The R Journal

Volume 13/2, December 2021

A peer-reviewed, open-access publication of the
R Foundation for Statistical Computing

Contents

Prospective authors will find detailed and up-to-date submission instructions on the Journal's homepage.

R Journal Homepage:
http://journal.r-project.org/

Email of editors and editorial board:
r-journal@R-project.org

The R Journal is indexed/abstracted by EBSCO, DOAJ, Thomson Reuters.

Editorial

by Dianne Cook

On behalf of the R Foundation and the Editorial board, I am pleased to present Volume 13 Issue 2 of the R Journal. This is the biggest issue ever!

First, some news from the Editorial board. A big thank you to Mike Kane, who has finished his term. As Editor-in-Chief in 2020, Mike expanded operations to include Associate Editors in the reviewing process. The R Journal now has a team of 20 Associate Editors. This has helped to manage the increasing number of submissions. We welcome new Associate Editors, Przemek Biecek, Chris Brunsdon, Mine Çetinkaya-Rundel, Kieran Healy, Adam Loy, Priyanga Dilini Talagala and Emi Tanaka, who have joined since the July 2021 issue. We also thank Taylor Arnold, who has stepped down, for his assistance over the past 18 months.

Catherine Hurley takes over as Editor-in-Chief for 2022, having joined the editorial team in 2020. She has substantial expertise in publishing research and editorial experience. One of the changes that she will oversee is publishing the R Journal **four** times a year. The benefit of this is that your articles will make a more timely appearance in an issue, and for us it will mean building slightly slimmer volumes. The issues will now be dated March, June, October and December, with articles that are accepted up to the publish month being included in the issue.

With the current issue we are migrating to the new web site. That is, the current dev `https://journal.r-project.org/dev/` will replace `https://journal.r-project.org/`. The current web site style is quite plain, and we would be very keen to get some feedback or help on the page design from users familiar with `css`. (Note that, the old site will remain as a legacy site.) This new site hosts articles in the new `html` format, if they have been written using Rmarkdown, as well as `pdf`. Please think about using the new Rmarkdown template for your article. It is quite pleasant to read, enables interactive graphics in the article, and the ability to add `alt-text` to your paper allows for screen readers to provide verbal descriptions of your figures for blind researchers. Also, there are simplified instructions for preparing an article to make it easier for you prepare your article for submission.

A few reminders about submissions. The journal operates purely through volunteer labour by researchers like yourself. We do what we can manage to get your paper in shape for publication, but ultimately what appears on the web is your responsibility.

- If you follow author instructions carefully, this will streamline the editorial board's handling of your article.
- Check that the references in your paper have all components, title, journal, volume, issue, pages, and the **DOI**.
- Check that the files you are about to submit do actually compile to the desirable output, every time prior to submitting your zip. Currently, there are easily a third of articles submitted that don't actually compile, and another third build with errors. This is time consuming to deal with when it comes time to build an issue.
- Check that your code is well-structured and runs in a timely fashion. A small example may be all that is necessary, in order to effectively communicate your work. Places to learn more about coding style are Jenny Bryan's Code Smells and Feels, and Hadley Wickham's Advanced R, and their collaborative book R Packages. Also, the discussion article and commentaries, in this issue, have excellent suggestions about developing your coding practices.
- All of the submitted files should be smallish. We use GitHub for journal operations, and files larger that 50Mb create complications for uploading. If you have a large data file, store it with one of the growing number of services for large files, such as figshare, dryad or zenodo. Provide links to these files in your example code, or in a section in the article listing supplementary material.
- When your zip file is downloaded into our paper handling system, a list of supple-

mentary files is automatically generated from what you report in the submission form, ideally. This list needs to be comma-delimited. When your paper is published this list of files is zip'd into a `supplementaries.zip` which is distributed on the issue web site. Journals do differ in what is distributed as supplementary. For the R Journal we would expect the list of files to include are `.R` file (R code) or `.Rmd`, any data files, and possibly an Appendix `pdf` or `html` if you want to communicate additional details like proofs or coding intricacies than were not possible to include in the paper.
- Choose at least one keyword from the list provided on the submission form, because these correspond to CRAN Task Views and helps connect your paper with other R developments. You can also type in keywords of your choosing as well.

The R Journal enjoys an increasing rank among statistics publications. It is a great outlet to publish your work. Statistical computing has a huge impact on the practice of statistics, and R Journal articles are a wonderful way to communicate your work in this area to a large audience. With the recent operational changes we are equipped to process a larger number of submissions. So make an impact, send us your work!

Lastly, there is a lot of work happening behind the scenes. Mitchell O'Hara-Wild continues to develop infrastructure. H. Sherry Zhang has spearheaded the changes to the rjtools package to help you, the authors, write your article in the style needed for the R Journal. The articles in this issue have been painstakingly copy edited by Dewi Amaliah. Funding from the R Consortium's has been instrumental in making all of these activities possible, and you can read more about it in the blog post here.

In this issue

News from the R Core, CRAN, Bioconductor, the R Foundation, and the foRwards Taskforce can be read in this issue.

This issue features 42 contributed research articles covering these topics, on a huge range of topics. There is also a special feature which is the discussion article "Software Engineering and R Programming: A Call for Research" by Melina Vidoni, and commentaries from Will Landau, Maëlle Salmon, Karthik Ram and Simon Urbanek.

Happy reading, and trying out the code!

Dianne Cook
Monash University

`https://journal.r-project.org`
`r-journal@r-project.org`

Software Engineering and R Programming: A Call for Research

by Melina Vidoni

Abstract Although R programming has been a part of research since its origins in the 1990s, few studies address scientific software development from a Software Engineering (SE) perspective. The past few years have seen unparalleled growth in the R community, and it is time to push the boundaries of SE research and R programming forwards. This paper discusses relevant studies that close this gap Additionally, it proposes a set of good practices derived from those findings aiming to act as a call-to-arms for both the R and RSE (Research SE) community to explore specific, interdisciplinary paths of research.

Introduction

R is a multi-paradigm statistical programming language, based on the S statistical language (Morandat et al., 2012), developed in the 1990s by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. It is maintained by the R Development Core Team (Thieme, 2018). Though CRAN (Comprehensive Archive Network) was created for users to suggest improvements and report bugs, nowadays, is the official venue to submit user-generated R packages (Ihaka, 2017). R has gained popularity for work related to statistical analysis and mathematical modelling and has been one of the fastest-growing programming languages (Muenchen, 2017). In July 2020, R ranked 8th in the TIOBE index, which measures of popularity of programming languages; as a comparison, one year before (July 2019), TIOBE ranked R in the 20th position (TIOBE, 2020). According to (Korkmaz et al., 2018), *"this has led to the development and distribution of over 10,000 packages, each with a specific purpose"*. Furthermore, it has a vibrant end-user programming community, where the majority of contributors and core members are *"not software engineers by trade, but statisticians and scientists"*, with diverse technical backgrounds and application areas (German et al., 2013).

R programming has become an essential part of *computational science*–the "application of computer science and Software Engineering (SE) principles to solving scientific problems" (Hasselbring et al., 2019). As a result, there are numerous papers discussing R packages explicitly developed to close a particular gap or assist in the analysis of data of a myriad of disciplines. Regardless of the language used, the development of software to assist in research ventures in a myriad of disciplines, has been termed as 'research SE' (RSE) (Cohen et al., 2021). Overall, RSE has several differences with traditional software development, such as the lifecycles used, the software goals and life-expectancy, and the requirements elicitation. This type of software is often "constructed for a particular project, and rarely maintained beyond this, leading to rapid decay, and frequent 'reinvention of the wheel" (Rosado de Souza et al., 2019).

However, both RSE and SE for R programming remain under-explored, with little SE-specific knowledge being tailored to these two areas. This poses several problems, given that in computational science, research software is a central asset for research. Moreover, although most RSE-ers (the academics writing software for research) come from the research community, a small number arrive from a professional programming background (Cohen et al., 2021; Pinto et al., 2018). Previous research showed R programmers do not consider themselves as developers (Pinto et al., 2018) and that few of them are aware of the intricacies of the language (Morandat et al., 2012). This poses a problem since the lack of formal programming training can lead to lower quality software (Hasselbring et al., 2019), as well as less-robust software (Vidoni, 2021a). This is problematic since ensuring sustainable development focused on code quality and maintenance is essential for the evolution of research in a myriad of computational science disciplines, as faulty and low-quality software can potentially affect research results (Cohen et al., 2018).

As a result, this paper aims to provide insights into three core areas:

- Related works that tackle both RSE and R programming, discussing their goals, motivations, relevancy, and findings. This list was curated through an unstructured review and is, by no means, complete or exhaustive.
- Organising the findings from those manuscripts into a list of good practices for developers. This is posed as a baseline, aiming to be improved with time, application, and experience.
- A call-to-arms for RSE and R communities, to explore interdisciplinary paths of research, covering not only empirical SE topics but also further developing the tools available to R programmers.

The rest of this paper is organised as follows. Section 2.2 presents the related works, introducing them one by one. Section 2.3 outlines the proposed best practices, and Section 2.4 concludes this work with a call-to-action for the community.

Related Works

This Section discusses relevant works organised in four sub-areas related to software development: coding in R, testing packages, reviewing them, and developers' experiences.

Area: Coding in R

Code quality is often related to technical debt. Technical Debt (TD) is a metaphor used to reflect the implied cost of additional rework caused by choosing an easy solution in the present, instead of using a better approach that would take longer (Samarthyam et al., 2017).

Claes et al. (2015) mined software repositories (MSR) to evaluate the *maintainability* of R packages published in CRAN. They focused on *function clones*, which is the practice of duplicating functions from other packages to reduce the number of dependencies; this is often done by copying the code of an external function directly into the package under development or by re-exporting the function under an alias. Code clones are harmful because they lead to redundancy due to code duplication and are a code smell (i.e., a practice that reduces code quality, making maintenance more difficult).

The authors identified that cloning, in CRAN packages only, is often caused by several reasons. These are: coexisting package versions (with some packages lines being cloned in the order of the hundreds and thousands), forked packages, packages that are cloned more than others, utility packages (i.e., those that bundle functions from other packages to simply importing), popular packages (with functions cloned more often than in other packages), and popular functions (specific functions being cloned by a large number of packages).

Moreover, they analysed the cloning trend for packages published in CRAN. They determined that the ratio of packages impacted by cloning appears to be stable but, overall, it represents over quarter-million code lines in CRAN. Quoting the authors, *"those lines are included in packages representing around 50% of all code lines in CRAN."* (Claes et al., 2015). Related to this, Korkmaz et al. (2019) found that the more dependencies a package has, the less likely it is to have a higher impact. Likewise, other studies have demonstrated that scantily updated packages that depend on others that are frequently updated are prone to have more errors caused by incompatible dependencies (Plakidas et al., 2017); thus, leading developers to clone functions rather than importing.

Code quality is also reflected by the comments developers write in their code. The notion of *Self-Admitted Technical Debt* (SATD) indicates the case where programmers are aware that the current implementation is not optimal and write comments alerting of the problems of the solution Potdar and Shihab (2014). Vidoni (2021b) conducted a three-part mixed-methods study to understand SATD in R programming, mining over 500 packages publicly available in GitHub and enquiring their developers through an anonymous online survey. Overall, this study uncovered that:

- Slightly more than 1/10th of the comments are actually "commenting out" (i.e., nullifying) functions and large portions of the code. This is a code smell named *dead code*, which represents functions or pieces of unused code that are never called or reached. It clogs the files, effectively reducing the readability Alves et al. (2016).
- About 3% of the source code comments are SATD, and 40% of those discuss code debt. Moreover, about 5% of this sample discussed *algorithm debt*, defined as *"sub-optimal implementations of algorithm logic in deep learning frameworks. Algorithm debt can pull down the performance of a system"* Liu et al. (2020).
- In the survey, developers declared adding SATD as "self reminders" or to "schedule future work", but also responded that they rarely address the SATD they encounter, even if it was added by themselves. This trend is aligned with what happens in traditional object-oriented (OO) software development.

This work extended previous findings obtained exclusively for OO, identifying specific debt instances as developers perceive them. However, a limitation of the findings is that the dataset was manually generated. For the moment, there is no tool or package providing support to detect SATD comments in R programming automatically.

Area: Testing R Packages

Vidoni (2021a) conducted a mixed-methods MSR (Mining Software Repositories) that combined mining GitHub repositories with a developers survey to study *testing technical debt* (TTD) in R programming–the test dimension of TD.

Overall, this study determined that R packages testing has poor quality, specifically caused by the situations summarised in Table 1. A finding is with regards to the type of tests being carried out. When designing test cases, good practices indicate that developers should test *common cases* (the "traditional" or "more used" path of an algorithm or function) as well as *edge cases* (values that require special handling, hence assessing boundary conditions of an algorithm or function) (Daka and Fraser, 2014). Nonetheless, this study found that almost 4/5 of the tests are common cases, and a vast majority of alternative paths (e.g., accessible after a condition) are not being assessed.

Moreover, this study also determined that available tools for testing are limited regarding their documentation and the examples provided (as indicated by survey respondents). This includes the usability of the provided assertions (given that most developers use custom-defined cases) and the lack of tools to automate the initialisation of data for testing, which often causes the test suits to fail due to problems in the suite itself.

Smell	Definition (Samarthyam et al., 2017)	Reason (Vidoni, 2021a)
Inadequate Unit Tests	The test suite is not ideal to ensure quality testing.	Many relevant lines remain untested. Alternative paths (i.e., those accessible after a condition) are mostly untested. There is a large variability in the coverage of packages from the same area (e.g., biostatistics). Developers focus on common cases only, leading to incomplete testing.
Obscure Unit Tests	When unit tests are obscure, it becomes difficult to understand the unit test code and the production code for which the tests are written.	Multiple asserts have unclear messages. Multiple asserts are mixed in the same test function. Excessive use of user-defined asserts instead of relying on the available tools.
Improper Asserts	Wrong or non-optimal usage of asserts leads to poor testing and debugging.	Testing concentrated on common cases. Excessive use of custom asserts. Developers still uncover bugs in their code even when the tests are passing.
Inexperienced Testers	Testers, and their domain knowledge, are the main strength of exploratory testing. Therefore, low tester fitness and non-uniform test accuracy over the whole system accumulate residual defects.	Survey participants are reportedly highly-experienced, yet their most common challenge was lack of testing knowledge and poor documentation of tools.
Limited Test Execution	Executing or running only a subset of tests to reduce the time required. It is a shortcut increasing the possibility of residual defects.	A large number of mined packages (about 35%) only used manual testing, with no automated suite (e.g., `testthat`). The survey responses confirmed this proportion.
Improper Test Design	Since the execution of all combination of test cases is an effort-intensive process, testers often run only known, less problematic tests (i.e., those less prone to make the system fail). This increases the risk of residual defects.	The study found a lack of support for automatically testing plots. The mined packages used `testthat` functions to generate a plot that was later (manually) inspected by a human to evaluate readability, suitability, and other subjective values. Survey results confirmed developers struggle with plots assessment.

Table 1: Problems found by Vidoni (2021a) regarding unit testing of R packages.

Křikava and Vitek (2018) conducted an MSR to inspect R packages' source code, making available a tool that automatically generates unit tests. In particular, they identified several challenges regarding testing caused by the language itself, namely its extreme dynamism, coerciveness, and lack of types, which difficult the efficacy of traditional test extraction techniques.

In particular, the authors worked with *execution traces*, "the sequence of operations performed by a program for a given set of input values" (Křikava and Vitek, 2018), to provide genthat, a package to optimise the unit testing of a target package (Krikava, 2018). **genthat** records the execution traces of a target package, allowing the extraction of unit test functions; however, this is limited to the public interface or the internal implementation of the target package. Overall, its process requires installation, extraction, tracing, checking and minimisation.

Both `genthat` and the study performed by these authors are highly valuable to the community since the *minimisation* phase of the package checks the unit tests and discards those failing, and records to coverage, eliminating redundant test cases. Albeit this is not a solution to the lack of edge cases detected in another study (Vidoni, 2021a), this **genthat** assists developers and can potentially reduce the workload required to obtain a baseline test suite. However, this work's main limitation is its emphasis on the coverage measure, which is not an accurate reflection of the tests' quality.

Finally, Russell et al. (2019) focused on the *maintainability quality* of R packages caused by their *testing* and *performance*. The authors conducted an MSR of 13500 CRAN packages, demonstrating that "reproducible and replicable software tests are frequently not available". This is also aligned with the findings of other authors mentioned in this Section. They concluded with recommendations to improve the long-term maintenance of a package in terms of testing and optimisation, reviewed in Section 2.3.

Area: Reviewing Packages

The increased relevance of software in data science, statistics and research increased the need for reproducible, quality-coded software (Howison and Herbsleb, 2011). Several community-led organisations were created to organize and review packages - among them, *rOpenSci* (Ram et al., 2019; rOpenSci et al., 2021) and *BioConductor* (Gentleman et al., 2004). In particular, *rOpenSci* has established a thorough peer-review process for R packages based on the intersection of academic peer-reviews and software reviews.

As a result, Codabux et al. (2021) studied *rOpenSci* open peer-review process. They extracted completed and accepted packages reviews, broke down individual comments, and performed a card sorting approach to determine which types of TD were most commonly discussed.

One of their main contributions is a taxonomy of TD extending the current definitions to R programming. It also groups debt types by *perspective*, representing 'who is the most affected by a type of debt". They also provided examples of rOpenSci's peer-review comments referring to a specific debt. This taxonomy is summarised in Table 2, also including recapped definitions.

Perspective	TD Type	Reason
User	Usability	In the context of R, test debt encompasses anything related to usability, interfaces, visualisation and so on.
	Documentation	For R, this is anything related to **roxygen2** (or alternatives such as the Latex or Markdown generation), readme files, vignettes and even **pkgdown** websites.
	Requirements	Refers to trade-offs made concerning what requirements the development team needs to implement or how to implement them.
Developer	Test	In the context of R, test debt encompasses anything related to coverage, unit testing, and test automation.
	Defect	Refers to known defects, usually identified by testing activities or by the user and reported on bug tracking systems.
	Design	For R, this debt is related to any OO feature, including visibility, internal functions, the triple-colon operator, placement of functions in files and folders, use of **roxygen2** for imports, returns of objects, and so on.
	Code	In the context of R, examples of code debt are anything related to renaming classes and functions, $<-$ vs. $=$, parameters and arguments in functions, FALSE/TRUE vs. F/T, print vs warning/message.
CRAN	Build	In the context of R, examples of build debt are anything related to Travis, Codecov.io, GitHub Actions, CI, AppVeyor, CRAN, CMD checks, `devtools::check`.
	Versioning	Refers to problems in source code versioning, such as unnecessary code forks.
	Architecture	for example, violation of modularity, which can affect architectural requirements (e.g., performance, robustness).

Table 2: Taxonomy of TD types and perspectives for R packages, proposed by Codabux et al. (2021).

Additionally, they uncovered that almost one-third of the debt discussed is *documentation debt*–related to how well packages are being documented. This was followed by *code debt*, providing a different distribution than the one obtained by Vidoni (2021b). This difference is caused by rOpenSci reviewers focusing on documentation (e.g., comments written by reviewers' account for most of the *documentation debt*), while developers' comments concentrate their attention in *code debt*. The entire classification process is detailed in the original study Codabux et al. (2021).

Area: Developers' Experiences

Developers' perspectives on their work are fundamental to understand how they develop software. However, scientific software developers have a different point of view than 'traditional' programmers (Howison and Herbsleb, 2011).

Pinto et al. (2018) used an online questionnaire to survey over 1500 R developers, with results enriched with metadata extracted from GitHub profiles (provided by the respondents in their answers). Overall, they found that scientific developers are primarily self-taught but still consider peer-learning a second valuable source. Interestingly, the participants did not perceive themselves as programmers, but rather as a member of any other discipline. This also aligns with findings provided by other works (German et al., 2013; Morandat et al., 2012). Though the latter is understandable, such perception may pose a risk to the development of quality software as developers may be inclined to feel 'justified' not to follow good coding practices Pinto et al. (2018).

Additionally, this study found that scientific developers work alone or in small teams (up to five people). Interestingly enough, they found that people spend a significant amount of time focused on coding and testing and performed an ad-hoc elicitation of requirements, mostly 'deciding by themselves' on what to work next, rather than following any development lifecycle.

When enquiring about commonly-faced challenges, the participants of this study considered the following: cross-platform compatibility, poor documentation (which is a central topic for reviewers (Codabux et al., 2021)), interruptions while coding, lack of time (also mentioned by developers in another study (Vidoni, 2021b)), scope bloat, lack of user feedback (also related to validation, instead of verification testing), and lack of formal reward system (e.g., the work is not credited in the scientific community (Howison and Herbsleb, 2011)).

Area	Main Problem	Recommended Practice
Lifecycles	The lack of proper requirement elicitation and development organisation was identified as a critical problem for developers (Wiese et al., 2020; Pinto et al., 2018), who often resort to writing comments in the source to remind themselves of tasks they later do not address (Vidoni, 2021b).	There are extremely lightweight agile lifecycles (e.g., Extreme Programming, Crystal Clear, Kanban) that can be adapted for a single developer or small groups. Using these can provide a project management framework that can also organise a research project that depends on creating scientific software.
Teaching	Most scientific developers do not perceive themselves as programmers and are self-taught (Pinto et al., 2018). This hinders their background knowledge and the tools they have available to detect TD and other problems, potentially leading to low-quality code (German et al., 2013).	Since graduate school is considered fundamental for these developers (Pinto et al., 2018), providing a solid foundation of SE-oriented R programming for candidates whose research relies heavily on software can prove beneficial. The topics to be taught should be carefully selected to keep them practical and relevant yet still valuable for the candidates.
Coding	Some problems discussed where functions clones, incorrect imports, non-semantic or meaningful names, improper visibilitiy or file distribution of functions, among others.	Avoid duplicating (i.e., copy-pasting or re-exporting) functions from other packages, and instead use proper selective import, such as **roxygen2**'s `@importFrom` or similar Latex documentation styles. Avoid leaving unused functions or pieces of code that are 'commented out' to be nullified. Proper use of version control enables developers to remove the segments of code and revisit them through previous commits. Code comments are meant to be meaningful and should not be used as a planning tool. Comments indicating problems or errors should be addressed (either when found, if the problem is small or planning for a specific time to do it if the problem is significant). Names should be semantic and meaningful, maintaining consistency in the whole project. Though there is no pre-established convention for R, previous works provide an overview (Baath, 2012), as well as packages, such as the tidyverse's style guide.
Testing	Current tests leave many relevant paths unexplored, often ignoring the testing of edge cases and damaging the robustness of the code packaged (Vidoni, 2021a; Russell et al., 2019)	All alternative paths should be tested (e.g., those limited by conditionals). Exceptional cases should be tested; e.g., evaluating that a function throws an exception or error when it should, and evaluating other cases such as (but not limited to), nulls, `NA`s, `NaN`s, warnings, large numbers, empty strings, empty variables (e.g., `character(0)`, among others. Other specific testing cases, including performance evaluation and profiling, discussed and exemplified by Russell et al. (2019).

Table 3: Recommendations of best practices, according to the issues found in previous work and good practices established in the SE community.

This study (Pinto et al., 2018) was followed up to create a taxonomy of problems commonly faced by scientific developers (Wiese et al., 2020). They worked with over 2100 qualitatively-reported problems and grouped them into three axes; given the size of their taxonomy, only the larger groups are summarised below:

- *Technical Problems:* represent almost two-thirds of the problems faced. They are related to software design and construction, software testing and debugging, software maintenance and evolution, software requirements and management, software build and release engineering, software tooling' and others (e.g., licensing, CRAN-related, user interfaces).
- *Social-Related Problems:* they represent a quarter of the problems faced by developers. The main groups are: publicity, lack of support, lack of time, emotional and communication and collaboration.
- *Scientific-Related Problems:* are the smaller category related to the science supporting or motivating the development. The main groups are: scope, background, reproducibility and data handling, with the latter being the most important.

These two works provide valuable insight into scientific software developers. Like other works mentioned in this article, albeit there are similarities with traditional software development (both in terms of programming paradigms and goals), the differences are notable enough to warrant further specialised investigations.

Towards Best Practices

Based on well-known practices for traditional software development (Sommerville, 2015), this Section outlines a proposal of best practices for R developers. These are meant to target the weaknesses found by the previous studies discussed in Section 2.2. This list aims to provide a baseline, aiming that (through future research works) they can be improved and further tailored to the needs of scientific software development and the R community in itself.

The practices discussed span from overarching (e.g., related to processes) to specific activities. They are summarised in Table 3.

Call to Action

Scientific software and R programming became ubiquitous to numerous disciplines, providing essential analysis tools that could not be completed otherwise. Albeit R developers are reportedly struggling in several areas, academic literature centred on the development of scientific software is scarce. As a result, this Section provides two calls to actions: one for R users and another for RSE academics.

Research Software Engineering Call: SE for data science and scientific software development is crucial for advancing research outcomes. As a result, interdisciplinary works are increasingly needed to approach specific areas. Some suggested topics to kickstart this research are as follows:

- *Lifecycles and methodologies for project management.* Current methodologies focus on the demands of projects with clear stakeholders and in teams of traditional developers. As suggested in Section 2.3, many agile methodologies are suitable for smaller teams or even uni-personal developments. Studying this and evaluating its application in practice can prove highly valuable.
- *Specific debts in scientific software.* Previous studies highlighted the existence of specific types of debt that are not often present in traditional software development (e.g., algorithm and reproducibility) (Liu et al., 2020) and are therefore not part of currently accepted taxonomies (Alves et al., 2016; Potdar and Shihab, 2014). Thus, exploring these specific problems can help detect uncovered problems, providing viable paths of actions and frameworks for programmers.
- *Distinct testing approaches.* R programming is an inherently different paradigm, and current guidance for testing has been developed for the OO paradigm. As a result, more studies are needed to tackle specific issues that may arise, such as how to test visualisations or scripts (Vidoni, 2021a), and how to move beyond coverage by providing tests that are optimal yet meaningful Křikava and Vitek (2018).

R Community Call: The following suggestions are centred on the abilities of the R community:

- Several packages remain under-developed, reportedly providing incomplete tools. This happens not only in terms of functionalities provided but also on their documentation and examples. For instance, developers disclosed that lack of specific examples was a major barrier when properly testing (Vidoni, 2021a). Extending the examples available in current packages can be achieved through community calls, leveraging community groups' reach, such as R-Ladies and RUGs (R User Groups). Note that this suggestion is not related to package development guides but to a community-sourced improvement of the documentation of existing packages.

- Additionally, incorporating courses in graduate school curricula that focus on "SE for Data Science" would be beneficial for the students, as reported in other works (Pinto et al., 2018; Wiese et al., 2020). However, this can only be achieved through interdisciplinary work that merges specific areas of interest with RSE academics and educators alike. Once more, streamlined versions of these workshops could be replicated in different community groups.

There is a wide range of possibilities and areas to work, all derived from diversifying R programming and RSE. This paper highlighted meaningful work in this area and proposed a call-to-action to further this area of research and work. However, these ideas need to be repeatedly evaluated and refined to be valuable to R users.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The author is grateful to both R-Ladies and rOpenSci communities that fostered the interest in this topic and to Prof. Dianne Cook for extending the invitation for this article.

Packages Mentioned

The following packages were mentioned in this article:

- **covr**, for package coverage evaluation. Mentioned by Křikava and Vitek (2018) and Codabux et al. (2021). Available at: https://cran.r-project.org/web/packages/covr/index.html.
- **genthat**, developed by Křikava and Vitek (2018), to optimise testing suits. Available at https://github.com/PRL-PRG/genthat.
- **pkgdown** for package documentation. Mentioned by Codabux et al. (2021) as part of documentation debt. Available at: https://cran.r-project.org/web/packages/pkgdown/index.html.
- **roxygen2**, for package documentation. Recommended in Section 2.3, and mentioned as examples of design and documentation debt by Codabux et al. (2021). Available at https://cran.r-project.org/web/packages/roxygen2/index.html.
- **testthat**, most used testing tool, according to findings by Vidoni (2021a). Mentioned when discussing testing debt by Codabux et al. (2021). Available at https://cran.r-project.org/web/packages/testthat/index.html.
- **tidyverse**, bundling a large number of packages and providing a style guile. Mentioned in Section 2.3. Available at: https://cran.r-project.org/web/packages/tidyverse/index.html.

Bibliography

N. S. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70:100–121, 2016. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2015.10.008. [p7, 11]

R. Baath. The state of naming conventions in r. *The R Journal*, 4:74–75, 12 2012. doi: 10.32614/RJ-2012-018. [p10]

M. Claes, T. Mens, N. Tabout, and P. Grosjean. An empirical study of identical function clones in CRAN. In *2015 IEEE 9th International Workshop on Software Clones (IWSC)*, pages 19–25, Mar. 2015. doi: 10.1109/IWSC.2015.7069885. [p7]

Z. Codabux, M. Vidoni, and F. Fard. Technical Debt in the Peer-Review Documentation of R Packages: a rOpenSci Case Study. In *2021 International Conference on Mining Software Repositories*, pages 1–11, Madrid, Spain, 2021. IEEE. doi: https://arxiv.org/abs/2103.09340. [p9, 10, 12]

J. Cohen, D. S. Katz, M. Barker, R. Haines, and N. Chue Hong. Building a sustainable structure for research software engineering activities. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 31–32, 2018. doi: 10.1109/eScience.2018.00015. [p6]

J. Cohen, D. S. Katz, M. Barker, N. Chue Hong, R. Haines, and C. Jay. The four pillars of research software engineering. *IEEE Software*, 38(1):97–105, 2021. doi: 10.1109/MS.2020.2973362. [p6]

E. Daka and G. Fraser. A survey on unit testing practices and problems. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pages 201–211, 2014. [p8]

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, Sep 2004. ISSN 1474-760X. doi: 10.1186/gb-2004-5-10-r80. URL https://doi.org/10.1186/gb-2004-5-10-r80. [p9]

D. M. German, B. Adams, and A. E. Hassan. The Evolution of the R Software Ecosystem. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 243–252, Mar. 2013. doi: 10.1109/CSMR.2013.33. ISSN: 1534-5351. [p6, 9, 10]

W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis. Fair and open computer science research software, 2019. [p6]

J. Howison and J. D. Herbsleb. Scientific software production: Incentives and collaboration. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, page 513–522, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305563. doi: 10.1145/1958824.1958904. URL https://doi.org/10.1145/1958824.1958904. [p9, 10]

R. Ihaka. The r project: A brief history and thoughts about the future, 2017. URL https://www.stat.auckland.ac.nz/~{}ihaka/downloads/Massey.pdf. [p6]

G. Korkmaz, C. Kelling, C. Robbins, and S. A. Keller. Modeling the Impact of R Packages Using Dependency and Contributor Networks. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 511–514, Aug. 2018. doi: 10.1109/ASONAM.2018.8508255. ISSN: 2473-991X. [p6]

G. Korkmaz, C. Kelling, C. Robbins, and S. Keller. Modeling the impact of Python and R packages using dependency and contributor networks. *Social Network Analysis and Mining*, 10(1):7, Dec. 2019. ISSN 1869-5469. doi: 10.1007/s13278-019-0619-1. URL https://doi.org/10.1007/s13278-019-0619-1. [p7]

F. Krikava. fikovnik/ISSTA18-artifact: ISSTA'18 Artifact release, July 2018. URL https://doi.org/10.5281/zenodo.1306437. [p8]

F. Křikava and J. Vitek. Tests from traces: Automated unit test extraction for r. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2018, page 232–241, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356992. doi: 10.1145/3213846.3213863. URL https://doi.org/10.1145/3213846.3213863. [p8, 11, 12]

J. Liu, Q. Huang, X. Xia, E. Shihab, D. Lo, and S. Li. Is Using Deep Learning Frameworks Free? Characterizing Technical Debt in Deep Learning Frameworks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, ICSE-SEIS '20, page 1–10, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371244. doi: 10.1145/3377815.3381377. URL https://doi.org/10.1145/3377815.3381377. [p7, 11]

F. Morandat, B. Hill, L. Osvald, and J. Vitek. Evaluating the Design of the R Language. In J. Noble, editor, *ECOOP 2012 – Object-Oriented Programming*, Lecture Notes in Computer Science, pages 104–131, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-31057-7. doi: 10.1007/978-3-642-31057-7_6. [p6, 9]

B. Muenchen. R's growth continues to accelerate, 2017. URL https://www.r-bloggers.com/rs-growth-continues-to-accelerate/. [p6]

G. Pinto, I. Wiese, and L. F. Dias. How do scientists develop scientific software? an external replication. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 582–591, 2018. doi: 10.1109/SANER.2018.8330263. [p6, 9, 10, 11, 12]

K. Plakidas, D. Schall, and U. Zdun. Evolution of the r software ecosystem: Metrics, relationships, and their impact on qualities. *Journal of Systems and Software*, 132:119–146, 2017. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2017.06.095. URL https://www.sciencedirect.com/science/article/pii/S0164121217301371. [p7]

A. Potdar and E. Shihab. An Exploratory Study on Self-Admitted Technical Debt. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 91–100, 2014. doi: 10.1109/ICSME.2014.31. [p7, 11]

K. Ram, C. Boettiger, S. Chamberlain, N. Ross, M. Salmon, and S. Butland. A community of practice around peer review for long-term research software sustainability. *Computing in Science Engineering*, 21(2):59–65, 2019. doi: 10.1109/MCSE.2018.2882753. [p9]

rOpenSci, B. Anderson, S. Chamberlain, L. DeCicco, J. Gustavsen, A. Krystalli, M. Lepore, L. Mullen, K. Ram, N. Ross, M. Salmon, and M. Vidoni. rOpenSci Packages: Development, Maintenance, and Peer Review, Feb. 2021. URL https://doi.org/10.5281/zenodo.4554776. [p9]

M. Rosado de Souza, R. Haines, M. Vigo, and C. Jay. What makes research software sustainable? an interview study with research software engineers. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 135–138, 2019. doi: 10.1109/CHASE.2019.00039. [p6]

S. Russell, T. D. Bennett, and D. Ghosh. Software engineering principles to improve quality and performance of R software. *PeerJ Computer Science*, 5:e175, Feb. 2019. ISSN 2376-5992. doi: 10.7717/peerj-cs.175. Publisher: PeerJ Inc. [p9, 10]

G. Samarthyam, M. Muralidharan, and R. K. Anna. *Understanding Test Debt*, pages 1–17. Springer Singapore, Singapore, 2017. ISBN 978-981-10-1415-4. doi: 10.1007/978-981-10-1415-4_1. URL https://doi.org/10.1007/978-981-10-1415-4_1. [p7, 8]

I. Sommerville. *Software Engineering*. Pearson, 10th edition, 2015. ISBN 0133943038. [p11]

N. Thieme. R generation. *Significance*, 15(4):14–19, 2018. doi: 10.1111/j.1740-9713.2018.01169.x. [p6]

TIOBE. Tiobe index - the software quality company, 2020. URL https://www.tiobe.com/tiobe-index/. [p6]

M. Vidoni. Evaluating unit testing practices in r packages. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, pages 1–12, Madrid, Spain, 2021a. IEEE. [p6, 8, 10, 11, 12]

M. Vidoni. Self-Admitted Technical Debt in R Packages: An Exploratory Study. In *2021 International Conference on Mining Software Repositories*, pages 1–11, Madrid, Spain, 2021b. IEEE. [p7, 9, 10]

I. Wiese, I. Polato, and G. Pinto. Naming the pain in developing scientific software. *IEEE Software*, 37(4):75–82, 2020. doi: 10.1109/MS.2019.2899838. [p10, 11, 12]

Melina Vidoni
Australian National University, School of Computing
Canberra, Australia
0000-0002-4099-1430
melina.vidoni@anu.edu.au

We Need Trustworthy R Packages

by William Michael Landau

Abstract There is a need for rigorous software engineering in R packages, and there is a need for new research to bridge scientific computing with more traditional computing. Automated tools, interdisciplinary graduate courses, code reviews, and a welcoming developer community will continue to democratize best practices. Democratized software engineering will improve the quality, correctness, and integrity of scientific software, and by extension, the disciplines that rely on it.

Commentary

Most contributors to R (R Core Team, 2021) do not see themselves as software engineers. In a way, this is part of the success of the language. As Dr. Vidoni explains, R developers are usually statisticians, economists, geneticists, ecologists, psychologists, sociologists, archaeologists, and other quantitative scientists who collectively pool a staggering diversity of academic knowledge into a cohesive repository of interoperable software packages. This rich ecosystem attracts a diverse user base and spurs popularity on a global scale.

But quality software still requires software engineering: specification, design, implementation, version control, testing, profiling, benchmarking, and documentation to produce packages worthy of trust. And because of its explosive adoption in recent decades, R needs trustworthy packages now more than ever. In the life sciences, for example, researchers increasingly use R to design, simulate, and analyze clinical trials (The R Foundation for Statistical Computing (2021), Nicholls et al. (2021), Gans et al. (2021), Wassmer and Pahlke (2021)). The resulting claims about safety and efficacy influence the medical treatments of millions of patients.

Fortunately, software engineering has begun to spread among self-described non-engineers. Workflow packages such as **testthat** (Wickham, 2011) and **covr** (Hester, 2020) identify essential but accessible practices and adapt them to an R-focused audience. On top of the popular packages that the article cites, newer specialized ones are under active development. One such example is **autotest** (Padgham, 2021), which automatically generates testing specifications that help developers identify uncommon boundary cases in statistical packages. Another is **valtools** (Hughes et al., 2021), a validation framework in which package developers declare formal requirements and explicitly map each requirement to one or more unit tests. **valtools**, part of the Pharmaceutical Users Software Exchange (PHUSE, Tinazzi et al. (2008)), was created to help R developers in the life sciences meet the requirements of regulatory authorities such as the United States Food and Drug Administration.

Still, key engineering issues remain underexplored for R, many of which fall outside the scope of the article. For example, what are the best ways to translate the logic of an algorithm into a collection of concise pure functions with sufficient encapsulation? Under what circumstances is it beneficial to clone an external function? (Claes et al. (2015) argue that cloning is not always harmful.) When is it appropriate to use ordinary functions, generic function object-oriented programming, e.g. S3 (Chambers, 2014), or more traditional message-passing OOP, e.g. **R6** (Chang, 2020)? Which design patterns are available for OOP and functional programming, how do they apply to R specifically, and which problems can they solve in real-life statistical modeling packages? When an anti-pattern is identified and classified, how can a technical debt taxonomy offer tailored recommendations for refactoring? How exactly does a package author write a specification to communicate the package's architecture and design principles to other developers? How do developers find optimal tradeoffs among automation, coverage, and computation time in unit testing?

As Dr. Vidoni points out, additional research may help translate long-established aspects of traditional software development to the world of R, and graduate courses may help instill this knowledge in new generations of quantitative scientists. Courses could borrow heavily from traditional computer science, especially the long history of object-oriented programming and functional programming. And just a little bit of exposure to a language like Haskell (Marlow, 2010), C++ (Stroustrup, 2013), Java (Gosling et al., 2015), or Python (Rossum, 1995) can help foster a well-rounded perspective. Even if students abandon these languages later on, they will retain pertinent concepts that R programmers seldom consciously utilize: for example, how immutable bindings serve as helpful guardrails in functional programming.

Code review, which the R community underutilizes, also aligns with the article's call to action. Reviews typically happen during a formal gatekeeping process, such as acceptance into CRAN (CRAN Volunteers, 2021), Bioconductor (Huber et al., 2015), or rOpenSci (Ram et al., 2019), or within small teams in order to expedite specific deliverables. There is usually a clear extrinsic need and a clearly identified expected extrinsic outcome. Pedagogical retrospectives are far less common, especially across different organizations, but they can be eye-opening experiences that substantially improve the awareness and capabilities of the mentees.

New social technologies could increase the frequency and effectiveness of code reviews and raise the collective software engineering skill level. For example, conferences in R, Statistics, and related fields could organize code review workshops, where mentees bring their own projects and experienced mentors provide in-person one-on-one feedback. In fact, entire R conferences could be dedicated to code review. Precedents include the Tidyverse Developer Day (Wickham et al., 2020) and the rOpenSci Unconference (rOpenSci, 2018), in which participants spend the majority of their time collaboratively working on code.

It is also possible to systematize an ongoing community-driven code review process for packages in public repositories. A fit-for-purpose public online forum could carry out ad hoc code reviews, and much like Stack Overflow, support a reward and reputation system for both mentors and mentees. A working group, possibly funded by the R Consortium (R Consortium, 2021) or similar, could kickstart the forum by selecting packages from CRAN, GitHub, etc. and inviting the authors to participate.

In summary, there is a need for rigorous software engineering in R packages, and there is a need for new research to bridge scientific computing with more traditional computing. Automated tools, interdisciplinary graduate courses, code reviews, and a welcoming developer community will continue to democratize best practices. Democratized software engineering will improve the quality, correctness, and integrity of scientific software, and by extension, the disciplines that rely on it.

Bibliography

J. Chambers. Object-Oriented Programming, Functional Programming and R. *Statistical Science*, 29, 09 2014. URL `https://doi.org/10.1214/13-STS452`. [p15]

W. Chang. *R6: Encapsulated Classes with Reference Semantics*, 2020. URL `https://CRAN.R-project.org/package=R6`. R package version 2.5.0. [p15]

M. Claes, T. Mens, N. Tabout, and P. Grosjean. An Empirical Study of Identical Function Clones in CRAN. *2015 IEEE 9th International Workshop on Software Clones, IWSC 2015 - Proceedings*, 03 2015. doi: https://doi.org/10.1109/IWSC.2015.7069885. [p15]

CRAN Volunteers. The Comprehensive R Archive Network, 2021. URL `https://cran.r-project.org/`. [p15]

M. Gans, A. Clark, R. Krajcik, M. Gotti, and N. Mockler. *tidyCDISC: tidyCDISC: Quick Exploratory Data Analyses on ADaM-ish Datasets*, 2021. URL `https://github.com/Biogen-Inc/tidyCDISC`. R package version 0.0.0.9000. [p15]

J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. The Java Language Specification, 2015. URL `https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf`. [p15]

J. Hester. *covr: Test Coverage for Packages*, 2020. URL `https://CRAN.R-project.org/package=covr`. R package version 3.5.1. [p15]

W. Huber, V. J. Carey, R. Gentleman, S. Anders, M. Carlson, B. S. Carvalho, H. C. Bravo, S. Davis, L. Gatto, T. Girke, R. Gottardo, F. Hahne, K. D. Hansen, R. A. Irizarry, M. Lawrence, M. I. Love, J. MacDonald, V. Obenchain, A. K. Ole's, H. Pag'es, A. Reyes, P. Shannon, G. K. Smyth, D. Tenenbaum, L. Waldron, and M. Morgan. Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*, 12(2):115–121, 2015. URL `https://doi.org/10.1038/nmeth.3252`. [p15]

E. Hughes, E. Miller, M. Vendettuoli, and P. Eshghi. *valtools: Automate Validated Package Creation*, 2021. URL `https://github.com/phuse-org/valtools`. [p15]

S. Marlow. Haskell 2010 Language Report, 2010. URL `https://www.haskell.org/onlinereport/haskell2010/`. [p15]

A. Nicholls, P. R. Bargo, and J. Sims. A risk-based approach for assessing R package accuracy within a validated infrastructure, 2021. URL `https://www.pharmar.org/white-paper/`. [p15]

M. Padgham. *autotest: Automatic Package Testing*, 2021. URL `https://docs.ropensci.org/autotest/`. [p15]

R Consortium. R Consortium, 2021. URL `https://www.r-consortium.org`. [p16]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p15]

K. Ram, C. Boettiger, S. Chamberlain, N. Ross, M. Salmon, and S. Butland. A Community of Practice Around Peer Review for Long-Term Research Software Sustainability. *Computing in Science and Engineering*, 21(2):59–65, 2019. URL https://doi.org/10.1109/MCSE.2018.2882753. [p15]

rOpenSci. rOpenSci Unconference 2018, 2018. URL https://unconf18.ropensci.org. [p16]

G. Rossum. Python Tutorial. Technical report, CWI (Centre for Mathematics and Computer Science), 1995. [p15]

B. Stroustrup. *The C++ Programming Language*. Addison-Wesley Professional, 4th edition, 2013. ISBN 0321563840. [p15]

The R Foundation for Statistical Computing. R: Regulatory Compliance and Validation Issues A Guidance Document for the Use of R in Regulated Clinical Trial Environments, 2021. URL https://www.r-project.org/doc/R-FDA.pdf. [p15]

A. Tinazzi, M. Scott, and A. Compagnoni. A 'systematic review' of PhUSE: what PhUSE users made available to the public after three years. *Pharmaceutical Programming*, 1(1):5–8, 2008. URL https://doi.org/10.1179/175709208X334597. [p15]

G. Wassmer and F. Pahlke. *rpact: Confirmatory Adaptive Clinical Trial Design and Analysis*, 2021. URL https://CRAN.R-project.org/package=rpact. R package version 3.1.0. [p15]

H. Wickham. testthat: Get Started with Testing. *The R Journal*, 3(1):5–10, 2011. URL https://doi.org/10.32614/RJ-2011-002. [p15]

H. Wickham, M. Averick, and J. Bryan. Tidyverse Developer Day, 2020. URL https://github.com/tidyverse/tidy-dev-day. [p16]

William Michael Landau
Eli Lilly and Company
893 Delaware St, Indianapolis, IN 46225
United States of America
ORCID: 0000-0003-1878-3253
will.landau@gmail.com
URL: https://wlandau.github.io

The R Developer Community Does Have a Strong Software Engineering Culture

by Maëlle Salmon and Karthik Ram

Abstract There is a strong software engineering culture in the R developer community. We recommend creating, updating and vetting packages as well as keeping up with community standards. We invite contributions to the rOpenSci project, where participants can gain experience that will shape their work and that of their peers.

Introduction

The R programming language was originally created for statisticians, by statisticians, but evolved over time to attract a "massive pool of talent that was previously untapped" (Hadley Wickham in Thieme (2018)). Despite the fact that most R users are academic researchers and business data analysts without a background in software engineering, we are witnessing a rapid rise in software engineering within the community. In this comment we spotlight recent progress in tooling, dissemination and support, including specific efforts led by the rOpenSci project. We hope that readers will take advantage of and participate in the tools and practices we describe.

The modern R package developer toolbox: user-friendlier, more comprehensive

The basic infrastructure for creating, building, installing, and checking packages has been in place since the early days of the R language. During this time (1998-2011), the barriers to entry were very high and access to support and Q&A for beginners were extremely limited. With the introduction of the **devtools** (Wickham et al., 2021b) package in 2011, the process of creating and updating packages became substantially easier. Documentation also became simpler to maintain. The **roxygen2** (Wickham et al., 2021a) package allowed developers to keep documentation in sync with changes in code, similar to the doxygen approach that was embraced in more mature languages. Combined with the rise in popularity of StackOverflow and the growth of rstats blogs, the number of packages on the Comprehensive R Archive Network (CRAN) skyrocketed from 400 new packages in 2010 to 1000 new packages by 2014. As of this writing, there are nearly 19k packages on CRAN.

For novices without substantial software engineer experience, the early testing frameworks were also difficult to use. With the release of **testthat** (Wickham, 2011), testing also became smoother. There are now several actively maintained testing frameworks such as **tinytest** (van der Loo, 2020); as well as testthat-compatible specialized tooling for testing database interactions (**dittodb** (Keane and Vargas, 2020)), web resources (**vcr** (Chamberlain, 2021)), **httptest** (Richardson, 2021), and **webfakes** (Csárdi, 2021) which enables the use of an embedded C/C++ web server for testing HTTP clients like **httr2** (Wickham, 2021)).

The testthat package has recently been improved with snapshot tests that make it possible to test plot outputs. The rOpenSci project has released **autotest** (Padgham, 2021), a package that supports automatic mutation testing.

Beyond checking for compliance with R CMD CHECK, several other packages such as **goodpractice** (Csárdi and Frick, 2018), **riskmetric** (R Validation Hub et al., 2021), rOpenSci's **pkgcheck** (Padgham and Salmon, 2021) check packages against a large list of actionable, community recommended best practices for software development. Collectively these tools allow domain researchers to release software packages that meet high standards for software engineering.

The development and testing ecosystem of R is rich and has sometimes borrowed successful implementations from other languages (e.g. the vcr R package is a port, i.e. translation to R, of the vcr Ruby gem; testthat snapshot tests were inspired by JS Jest[1]).

Emergence of a welcoming community

As underlined in Thieme (2018), community is the strong suit of the R language. Many organizations and venues offer dedicated support for package developers. Examples include Q&A on the r-package-devel mailing list[2], and the package development category of the RStudio community forum[3], and

[1] https://www.tidyverse.org/blog/2020/10/testthat-3-0-0/#snapshot-testing
[2] https://stat.ethz.ch/mailman/listinfo/r-package-devel
[3] https://community.rstudio.com/c/package-development/11

the rstats section of StackOverflow[4]. Traditionally, R package developers have been mostly male and white. Although the status quo remains similar, efforts from groups such as R-Ladies[5] meetups, Minorities in R (Scott and Smalls-Perkins, 2020), and the package development modules offered by Forwards for underrepresented groups[6] have made considerable inroads towards improving diversity. These efforts have worked hard to put the spotlight on developers beyond the "usual suspects".

rOpenSci community and software review

The rOpenSci organization (Boettiger et al., 2015) is an attractive venue for developers & supporters of scientific R software. One of our most successful and continuing initiatives is our Software Peer Review system (Ram et al., 2019), a combination of academic peer-review and code review from industry. About 150 packages have been reviewed by volunteers to date, creating better packages as well as a growing knowledgebase in our development guide (rOpenSci et al., 2021) while also building a living community of practice.
Our model has been the fundamental inspiration for projects such as the Journal of Open Source Software (Smith et al., 2018), and PyOpenSci [Wasser and Holdgraf (2019)](Trizna et al., 2021). We are continuously improving our system and reducing cognitive overload on editors and reviewers by automating repetitive tasks. Most recently we have expanded our offerings to peer review of packages that implement statistical methods (Statistical Software Peer Review) (Padgham et al., 2021).
Beside software review, rOpenSci community is a safe, welcoming and informative place for package developers, with Q&A happening on our public forum and semi-open Slack workspace. (Butland and LaZerte, 2020)

Creation and dissemination of resources for R programmers

The aforementioned tools, venues and organizations benefit from and support crucial dissemination efforts.
Publishing technical know-how is crucial for progress of the R community. R news has been circulating on Twitter[7], R Weekly[8] and R-Bloggers[9]. Some sources have been more specifically aimed at R package developers of various experience and interests. While "Writing R Extensions" [10] is the official & exhaustive reference on writing R packages, it is a reference rather than a learning resource: many R package developers, if not learning by example, get introduced to R package development via introductory blog posts or tutorials, and the R packages book by Hadley Wickham and Jenny Bryan [Wickham (2015)](Wickham and Bryan) that accompany the devtools suite of packages is freely available online and strives to improving the R package development experience. The rOpenSci guide "rOpenSci Packages: Development, Maintenance, and Peer Review" (rOpenSci et al., 2021) contains our community-contributed guidance on how to develop packages and review them. It features *opinionated requirements* such as the use of **roxygen2** (Wickham et al., 2021a) for package documentation; *criteria helping make an informed decision* on gray area topics such as limiting dependencies; *advice on widely accepted and emerging tools*. As it is a living document also used as reference for editorial decisions, we maintain a changelog[11], and summarize each release in a blog post[12]. rOpenSci also hosts a book on a specialized topic, HTTP testing in R[13], that presents both principles for testing packages that interact with web resources, as well as relevant packages. Beside these examples of long-form documentation, knowledge around R software engineering is shared through blogs and talks. In the R blogging world, the rOpenSci blog posts[14], technical notes[15] and a section of our monthly newsletter[16] feature some topics relevant to package developers, as do some of the posts on the Tidyverse blog[17]. The blog of the R-hub project[18] contains information on package development topics, in particular about common problems such as sharing data via R packages or understanding CRAN checks. Expert programmers

[4]https://stackoverflow.com/questions/tagged/r?tab=Newest
[5]http://rladies.org/
[6]https://buzzrbeeline.blog/2021/02/09/r-forwards-package-development-modules-for-women-and-other-underrepresented-groups/
[7]https://www.t4rstats.com/
[8]https://rweekly.org/
[9]https://www.r-bloggers.com/
[10]https://cran.r-project.org/doc/manuals/R-exts.html
[11]https://devguide.ropensci.org/booknews.html
[12]https://ropensci.org/tags/dev-guide/
[13]https://books.ropensci.org/http-testing/
[14]https://ropensci.org/blog/
[15]https://ropensci.org/technotes/
[16]https://ropensci.org/news/
[17]https://www.tidyverse.org/categories/programming/
[18]https://blog.r-hub.io/post/

have been sharing their R specific wisdom as well as software engineering lessons learned from other languages (e.g. Jenny Bryan's useR! Keynote address "code feels, code smells"[19]).

Conclusion

In summary, we observe that there is already a strong software engineering culture in the R developer community. By surfacing the rich suite of resources to new developers we can but only hope the future will bring success to all aforementioned initiatives. We recommend creating, updating and vetting packages with the tools we mentioned as well as keeping up with community standards with the venues we mentioned in the previous section. We invite contributions to the rOpenSci project, where participants can gain experience that will shape their work and that of their peers. Thanks to these efforts, we hope the R community will continue to be a thriving place of application for software engineering, by diverse practitioners from many different paths.

Bibliography

C. Boettiger, S. Chamberlain, E. Hart, and K. Ram. Building software, building community: Lessons from the ropensci project. *Journal of Open Research Software*, 3(1):e8, 2015. doi: 10.5334/jors.bu. [p19]

S. Butland and S. LaZerte. *rOpenSci Community Contributing Guide*. Zenodo, 2020. doi: 10.5281/ZENODO.4000532. URL https://contributing.ropensci.org/. [p19]

S. Chamberlain. *vcr: Record 'HTTP' Calls to Disk*, 2021. URL https://CRAN.R-project.org/package=vcr. R package version 1.0.2. [p18]

G. Csárdi. *webfakes: Fake Web Apps for HTTP Testing*, 2021. https://webfakes.r-lib.org/, https://github.com/r-lib/webfakes. [p18]

G. Csárdi and H. Frick. *goodpractice: Advice on R Package Building*, 2018. URL https://CRAN.R-project.org/package=goodpractice. R package version 1.0.2. [p18]

J. Keane and M. Vargas. *dittodb: A Test Environment for Database Requests*, 2020. URL https://CRAN.R-project.org/package=dittodb. R package version 0.1.3. [p18]

M. Padgham. *autotest: Automatic Package Testing*, 2021. https://docs.ropensci.org/autotest/, https://github.com/ropensci-review-tools/autotest. [p18]

M. Padgham and M. Salmon. *pkgcheck: rOpenSci Package Checks*, 2021. https://docs.ropensci.org/pkgcheck/, https://github.com/ropensci-review-tools/pkgcheck. [p18]

M. Padgham, M. Salmon, N. Ross, J. Nowosad, R. FitzJohn, yilong zhang, C. Sax, F. Rodriguez-Sanchez, F. Briatte, and L. Collado-Torres. ropensci/statistical-software-review-book: Official first standards versions, Oct. 2021. URL https://doi.org/10.5281/zenodo.5556756. [p19]

R Validation Hub, D. Kelkhoff, M. Gotti, E. Miller, K. K, Y. Zhang, E. Milliman, and J. Manitz. *riskmetric: Risk Metrics to Evaluating R Packages*, 2021. https://pharmar.github.io/riskmetric/, https://github.com/pharmaR/riskmetric. [p18]

K. Ram, C. Boettiger, S. Chamberlain, N. Ross, M. Salmon, and S. Butland. A community of practice around peer review for long-term research software sustainability. *Computing in Science Engineering*, 21(2):59–65, 2019. doi: 10.1109/MCSE.2018.2882753. [p19]

N. Richardson. *httptest: A Test Environment for HTTP Requests*, 2021. https://enpiar.com/r/httptest/, https://github.com/nealrichardson/httptest. [p18]

rOpenSci, B. Anderson, S. Chamberlain, L. DeCicco, J. Gustavsen, A. Krystalli, M. Lepore, L. Mullen, K. Ram, N. Ross, M. Salmon, and M. Vidoni. rOpenSci Packages: Development, Maintenance, and Peer Review, Feb. 2021. URL https://doi.org/10.5281/zenodo.4554776. [p19]

D. Scott and D. Smalls-Perkins. Introducing mir: A community for underrepresented minority users of r, Feb 2020. URL https://medium.com/@doritolay/introducing-mir-a-community-for-underrepresented-users-of-r-7560def7d861. [p19]

[19]https://github.com/jennybc/code-smells-and-feels

A. M. Smith, K. E. Niemeyer, D. S. Katz, L. A. Barba, G. Githinji, M. Gymrek, K. D. Huff, C. R. Madan, A. C. Mayes, K. M. Moerman, P. Prins, K. Ram, A. Rokem, T. K. Teal, R. V. Guimera, and J. T. Vanderplas. Journal of open source software (JOSS): design and first-year review. *PeerJ Computer Science*, 4:e147, Feb. 2018. doi: 10.7717/peerj-cs.147. URL https://doi.org/10.7717/peerj-cs.147. [p19]

N. Thieme. R generation. *Significance*, 15(4):14–19, 2018. doi: 10.1111/j.1740-9713.2018.01169.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1740-9713.2018.01169.x. [p18]

M. Trizna, L. A Wasser, and D. Nicholson. pyopensci: Open and reproducible research, powered by python. *Biodiversity Information Science and Standards*, 5:e75688, 2021. doi: 10.3897/biss.5.75688. URL https://doi.org/10.3897/biss.5.75688. [p19]

M. van der Loo. A method for deriving information from running r code. *The R Journal*, page Accepted for publication, 2020. URL https://arxiv.org/abs/2002.07472. [p18]

L. A. Wasser and C. Holdgraf. pyOpenSci Promoting Open Source Python Software To Support Open Reproducible Science. In *AGU Fall Meeting Abstracts*, volume 2019, pages NS21A–13, Dec. 2019. [p19]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p18]

H. Wickham. *R Packages*. O'Reilly Media, 2015. [p19]

H. Wickham. *httr2: Perform HTTP Requests and Process the Responses*, 2021. URL https://CRAN.R-project.org/package=httr2. R package version 0.1.1. [p18]

H. Wickham and J. Bryan. R packages. Second Edition [Online]. URL https://r-pkgs.org/. [p19]

H. Wickham, P. Danenberg, G. Csárdi, and M. Eugster. *roxygen2: In-Line Documentation for R*, 2021a. URL https://CRAN.R-project.org/package=roxygen2. R package version 7.1.2. [p18, 19]

H. Wickham, J. Hester, and W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2021b. URL https://CRAN.R-project.org/package=devtools. R package version 2.4.2. [p18]

Maëlle Salmon
The rOpenSci Project

https://masalmon.eu
ORCiD: 0000-0002-2815-0399
msmaellesalmon@gmail.com

Karthik Ram
Berkeley Institute for Data Science and The rOpenSci Project

https://ram.berkeley.edu/
ORCiD: 0000-0002-0233-1757
karthik.ram@berkeley.edu

The R Quest: from Users to Developers

by Simon Urbanek

Abstract R is not a programming language, and this produces the inherent dichotomy between analytics and software engineering. With the emergence of data science, the opportunity exists to bridge this gap, especially through teaching practices.

Genesis: How did we get here?

The article "Software Engineering and R Programming: A Call to Action" summarizes the dichotomy between analytics and software engineering in the R ecosystem, provides examples where this leads to problems and proposes what we as R users can do to bridge the gap.

Data Analytic Language

The fundamental basis of the dichotomy is inherent in the evolution of S and R: they are not programming languages, but they ended up being mistaken for such. S was designed to be a *data analytic* language: to turn ideas into software quickly and faithfully, often used in "non-programming" style (Chambers, 1998). Its original goal was to enable the statisticians to apply code which was written in programming languages (at the time mostly FORTRAN) to analyze data quickly and interactively - for some suitable definition of "interactive" at the time (Becker, 1994). The success of S and then R can be traced to the ability to perform data analysis by applying existing tools to data in creative ways. A data analysis is a quest - at every step we learn more about the data which informs our decision about next steps. Whether it is an exploratory data analysis leveraging graphics or computing statistics or fitting models - the final goal is typically not known ahead of time, it is obtained by an iterative process of applying tools that we as analysts think may lead us further (Tukey, 1977). It is important to note that this is exactly the opposite of software engineering where there is a well-defined goal: a specification or desired outcome, which simply needs to be expressed in a way understandable to the computer.

Freedom for All

The second important design aspect rooted in the creativity required is the freedom the language provides. Given that the language can be computed upon means that a given expression may have different meaning depending on how the called function decides to treat it and such deviations are not entirely uncommon, typically referred to as non-standard evaluation. Probably the best example is the sub-language defined by the **data.table** package (Dowle and Srinivasan, 2021) featuring the := operator which is parsed, but not even used by the R language.

Analogously, there is no specific, prescribed object system, but rather one is free to implement any idea desirable, as witnessed by the fact that there are more than a handful of object system definitions available in R and contributed packages. This freedom is what makes R great for experimentation with new ideas or concepts, but very hard to treat as a programming language.

We have a language that is built on the idea of applying tools and which allows freedom to express new ideas so the last important step is how to define new tools. R add-on packages (R Core Team, 2021) are the vehicle by which new tools can be defined and distributed to R users. Note that true to design goals, packages are not limited to R code but rather can also include code written in programming languages such as C, C++ or Fortran. That in turn makes it possible to write packages that expand the scope of tools to other languages such as Java with **RJava** (Urbanek, 2021) or Python with **reticulate** (Ushey et al., 2022) simply by creating an R package which defines the interface.

Sharing Packages

But this is also where we are entering the realm of software engineering. Now we are in the business of *defining* the tools as opposed to just *using* the tools. It also means that the tools have to worry about programming interfaces, defining behavior and all those pesky things we as statisticians don't want to worry about. Although we originally started as R *users*, the moment we want to share any re-usable piece of code with others we are becoming *developers*. Since no developer would mistake R for a programming language, it is analysts with background in various fields which use statistics one way or another that are more likely to *use* R. However, as we become more comfortable with R, we start using it as a programming language, not just analytic language, often because it is simply more convenient than having to learn a programming language. This explains the empirical evidence (Pinto et al., 2018) of R package authors not being trained software engineers, but often scientists from other fields and any consequences thereof.

However, as R packages started to emerge, it became clear that a loosely coupled structure is not enough and have to introduce software engineering concepts such as documentation and testing. R includes tools for automated checking for packages to be able to provide at least some basic guarantees. Packages provide examples which are supposed to be illustrative, but soon were used to perform limited testing. R itself is using the same package structure and it was clear early that a test suite is important and so was introduced. Consequently, the same facilities were available to packages, but only very few were using it. There are, however, no built-in tools for creating test suites. In core R those are hand-curated by experienced developers, but that does not scale to package space.

Over 18,000 packages are now present in the Comprehensive R Archive Network (CRAN), a repository which has arguably played major role in the success of R (Hornik and Leisch, 2002). This is not only a valuable resource for users, but today this rich collection of contributed R code in being used as an automated test-suite for R. This is no coincidence, the importance of software engineering concepts has been identified by the CRAN team long time ago and the tools in R have been enhanced for that purpose (Hornik, 2016). CRAN has been an invaluable asset for the development of R based on examples and limited tests alone. It allows us the R Core Team to test changes in R against code that was written by ingenious people that do not necessarily follow documentation, but instead write code that seems to work - possibly in ways not intended in the first place. Consequently, improving the quality and coverage of tests in packages has not only positive impact on the individual package, but on the quality of the entire CRAN ecosystem and R itself.

CRAN performs reverse-dependency checks where packages are not allowed to break dependent package which is an important software engineering concept. One can see CRAN as performing continuous integration and continuous testing if we consider all submitted packages as one big project. This is not universally liked among package authors, though. Some find it too tedious to be responsible for software in the way a software engineer would be - a concern which is also highlighted by the article.

Steal and Borrow

One perhaps surprising finding of the article was the analysis of code fragment re-use (Claes et al., 2015). A quite recent example how dangerous such practice is was a piece of badly written JavaScript code from Stack Overflow (StackOverflow) which was copied so often that it made it into the popular Unity game engine, effectively forcing browsers to lie about macOS versions (Chromium Bugs) just to not break millions of released products. R code fragments are less likely to have such world-wide impact, but can be equally frustrating. The historically relatively high cost of loading other packages was an incentive to simply copy fragments instead, but the performance impact has been diminishing with advancements in the R implementation. Still, I believe the exact reasons for fragment re-use deserve further examination and may reveal other, more benign motives.

Every Project Needs a Conductor

Another good example of introducing software engineering principles into the R world successfully is the Bioconductor project (Gentleman et al., 2004). The authors realized early that the project is too big for it to allow organic growth and have strongly encouraged the use of the S4 class system to build a class hierarchy specific to the tasks common to the Bioconductor packages. This enabled optimizations of implementation as a core part of the system as opposed to individual approaches in each package. Bioconductor was also encouraging unit tests and has maintained a build and reporting system similar to that of CRAN, in the early days even pioneering functionality that was later added to core R.

The Gospel of Data Science

I believe the Call to Action is a very timely contribution. Many R users start as statisticians or data analysts in some domain since that is the main strength of R. Consequently, a lot of R code is never publicly visible. Code written for data analyses is not software development and is not published as software. So any global statistics about R code have to be taken with that in mind. When considering R packages we are talking only about a fraction of the code written in R. However, building new tools is an important part of the R ecosystem and it has to be made clear that it is different from data analysis and thus requires different skills and tools.

The main realization here is that at some point an R user may become an R developer, crossing the line from analysis into software engineering. And we are often unprepared for that, in part because of our diverse background. When I asked my junior colleagues at the Labs what they find most challenging yet valuable, the top item was learning software engineering skills on the job. We were lucky to have both the authors of S as well as the authors of Unix on the same floor, so we were able to bridge the gap, but generally our schools don't prepare for that. That's why I believe we must teach statistical computing together with software engineering skills such as re-usability and testing concepts. The current popularity of data science which bridges both worlds is a good excuse to make it actually happen in practice.

Bibliography

R. A. Becker. A brief history of S. Technical report, AT&T Bell Laboratories, 11 1994. [p22]

J. M. Chambers. *Programming with Data: A Guide to the S Language*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 1998. ISBN 0387985034. [p22]

Chromium Bugs. Nearly all Unity WebGL games fail to run in Chrome on macos 11 because of userAgent. URL https://bugs.chromium.org/p/chromium/issues/detail?id=1171998. [p23]

M. Claes, T. Mens, N. Tabout, and P. Grosjean. An empirical study of identical function clones in CRAN. In *2015 IEEE 9th International Workshop on Software Clones (IWSC)*, pages 19–25, Mar. 2015. doi: 10.1109/IWSC.2015.7069885. [p23]

M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2021. URL https://CRAN.R-project.org/package=data.table. R package version 1.14.2. [p22]

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, Sep 2004. ISSN 1474-760X. doi: 10.1186/gb-2004-5-10-r80. URL https://doi.org/10.1186/gb-2004-5-10-r80. [p23]

K. Hornik. Are there too many R packages? *Austrian Journal of Statistics*, 41(1):59–66, 2 2016. doi: 10.17713/ajs.v41i1.188. [p23]

K. Hornik and F. Leisch. Vienna and R: Love, marriage and the future. *Festschrift 50 Jahre Österreichische Statistische Gesellschaft*, pages 61–70, 01 2002. ISSN 2016-597X. [p23]

G. Pinto, I. Wiese, and L. F. Dias. How do scientists develop scientific software? An external replication. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 582–591, 2018. doi: 10.1109/SANER.2018.8330263. [p23]

R Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://cran.r-project.org/doc/manuals/r-release/R-exts.html. [p22]

StackOverflow. How to find the operating system details using JavaScript. URL https://stackoverflow.com/questions/9514179/how-to-find-the-operating-system-details-using-javascript. [p23]

J. W. Tukey. *Exploratory Data Analysis*. Number v. 2 in Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977. ISBN 9780201076165. [p22]

S. Urbanek. *rJava: Low-Level R to Java Interface*, 2021. URL https://CRAN.R-project.org/package=rJava. R package version 1.0-6. [p22]

K. Ushey, J. Allaire, and Y. Tang. *reticulate: Interface to Python*, 2022. URL https://CRAN.R-project.org/package=reticulate. R package version 1.23. [p22]

Simon Urbanek
University of Auckland
Department of Statistics
Auckland, New Zealand
urbanek@R-project.org

Rejoinder: Software Engineering and R Programming

by Melina Vidoni

Abstract It is a pleasure to take part in such fruitful discussion about the relationship between Software Engineering and R programming, and what could be gain by allowing each to look more closely at the other. Several discussants make valuable arguments that ought to be further discussed.

The Roles

It is worth arguing about the difference between **research software engineers** and **software engineering researchers**. While the former can be anyone developing scientific software for computation/data sciences (regardless of their technical background or "home" discipline), the latter are academics investigating software engineering in different domains.

Software engineering researchers aim to produce research that is translatable and usable by practitioners, and when investigating R programming (or any other type of scientific software) the "practitioners" are *research software engineers*. This distinction is relevant as one cannot work without the other. In other words, *software engineering researchers* ought to study *research software engineers* such like they study, e.g., a web developer, with the goal of uncovering their "pain points" and propose a solution to it. Likewise, *research software engineers* depend on *software engineering researchers* and expect them to produce the new knowledge they need.

However, what a *research software engineer* will vary by the programming language they use, and what they aim to achieve with it. In terms of R programming, as one discussant pointed, there can be a difference between an "R user" (which *uses* R to perform data analysis) and an "R developer" (which besides *using* the language, also *develops* it by creating publicly shared packages). However, to this extent, research has used both terms interchangeably, which leads to a possible avenue of work in terms of "human aspects of R programming".

The Software

This is where the next link appears–the **tools and packages** mentioned in the commentaries were developed with the intention of translating/migrating knowledge acquired/produced by *software engineering researchers* to the domain of R programming, and to be used by *research software engineers*. For example, the package `covr` streamlines the process of calculating the unit testing coverage of a package, and the original papers presenting such measures can be tracked down to the late "80s (Frankl and Weyuker, 1988; DeMillo, 1987). Albeit it is known coverage as a measure evolved and changed over time (and continues to do so), it is an excellent example of the outcome produced by *software engineering researchers* that successfully translated their findings to "practitioners" (in this case, *research software engineers*).

Therefore, a package is part of the "translation" of the knowledge acquired through software engineering research, into an accessible, usable framework. However, the tool itself is not enough–without the "environment" changing, growing, and learning, the tool may not be used to its full potential. Note that "environment" is used to refer (widely and loosely) to a person's programming habits, acceptance to change, past experiences (e.g., time/effort spent in solving a bug, or domains worked on), and even the people around them (e.g., doing/not doing something because of what others do/do not do) that influence their vision, attitude and expectations regarding programming.

Moreover tools and packages are not finite, static elements–because they are software, they evolve. And when the requirements of a community change, so must do so the tools. This act as a reminder to not assign a "silver bullet" status to a tool meant to solve a particular, static problem, when it has been known that software (and thus the practices to develop it) evolve, and may even become unmanageable, never to be fully solved (Brooks, 1987).

The Goal

Another related aspect is that "scientific software" has broader, different goals than "traditional" (namely, non-scientific) software development–it has been argued that "scientific software development" is concerned with knowledge acquisition rather than software production (Kelly, 2015); e.g. a "tool" can be an RMarkdown document that allows performing an analysis (hence, *using* the language).

Related to this, "scientific software" uses diverse paradigms, such as *literate programming* (which has been considered a programming paradigm for a few decades (Cordes and Brown, 1991)) and *scripting* (which in turn, continues to elicit mixed stances from *software engineering researchers* (Loui, 2008)) with goals different to "traditional software".

Thus, what "software engineering practices" mean for "scientific software" remains ambiguous, and some authors have argued that the "gap" between software engineering and scientific programming threatens the production of reliable scientific results (Storer, 2017). The following are some example questions meant to illustrate how these other aspects of "scientific software" may still be related to software engineering practices:

> *Could text in a literate programming file be considered documentation? Is scripting subjected to code-smell practices like incorrect naming or code reuse? Does self-admitted technical debt exists in literate/scripting programming? What is the usability of a literate programming document? Should analytical scripts be meant for reuse?*

The original article was intended to highlight some of the efforts made by *software engineering researchers* to bridge this gap of software engineering knowledge for "scientific programming". Nonetheless, *software engineering researchers* have perhaps focused more strongly on R packages because of their similarities to their current research (namely, "traditional software" development), thus making the translation of knowledge slightly more straightforward. Approaching other aspects, paradigms, tools and process of "scientific software" development still remains a gap on research that should be further studied.

The Community

The **community** is the next link in this chain–they motivate *software engineering researchers*" investigations, are the subjects, and the beneficiaries. Yet many times, they can also be the cause of their own "pain points". For example, research has shown that although StackOverflow is nowadays a staple for any programmer, many solutions derived from it can be outright insecure (Rahman et al., 2019; Fischer et al., 2017; Acar et al., 2016), have poor quality and code smells (Zhang et al., 2018; Meldrum et al., 2020), be outdated (Zhang, 2020; Zerouali et al., 2021), or have low performance (Toro, 2021), among others. This is but a facet of the concept of "there is no silver bullet" (Brooks, 1987), and the only way of solving such situation (partially, and temporarily) is to look at it from multiple points of views. This action is what the original paper aimed to highlight.

Final words

In the end, the differences between *software engineering researchers* and *research software engineers* are blurry, and the translation of concepts from "traditional software" development/research to "scientific software" development/research may not be as straightforward as both groups of stakeholders consider. However, for the R community to continue evolving, both can (and should) work together and learn from the other.

Bibliography

Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky. You get where you're looking for: The impact of information sources on code security. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 289–305, 2016. doi: 10.1109/SP.2016.25. [p26]

F. Brooks, Jr. No silver bullet essence and accidents of software engineering. *IEEE Computer*, 20:10–19, 04 1987. doi: 10.1109/MC.1987.1663532. [p25, 26]

D. Cordes and M. Brown. The literate-programming paradigm. *Computer*, 24(6):52–61, 1991. doi: 10.1109/2.86838. [p26]

R. A. DeMillo. *Software Testing and Evaluation*. Menlo Park, Calif Benjamin/Cummings Pub. Co, 1987. ISBN 978-0-8053-2535-5. [p25]

F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl. Stack overflow considered harmful? the impact of copy amp;paste on android application security. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 121–136, 2017. doi: 10.1109/SP.2017.31. [p26]

P. Frankl and E. Weyuker. An applicable family of data flow testing criteria. *IEEE Transactions on Software Engineering*, 14(10):1483–1498, 1988. doi: 10.1109/32.6194. [p25]

D. Kelly. Scientific software development viewed as knowledge acquisition. *Journal of Systems and Software*, 109(C): 50–61, nov 2015. ISSN 0164-1212. doi: 10.1016/j.jss.2015.07.027. URL https://doi.org/10.1016/j.jss.2015.07.027. [p25]

R. P. Loui. In praise of scripting: Real programming pragmatism. *Computer*, 41(7):22–26, 2008. doi: 10.1109/MC.2008.228. [p26]

S. Meldrum, S. A. Licorish, C. A. Owen, and B. T. R. Savarimuthu. Understanding stack overflow code quality: A recommendation of caution. *Science of Computer Programming*, 199:102516, 2020. ISSN 0167-6423. doi: https://doi.org/10.1016/j.scico.2020.102516. URL https://www.sciencedirect.com/science/article/pii/S0167642320301246. [p26]

A. Rahman, E. Farhana, and N. Imtiaz. Snakes in paradise?: Insecure python-related coding practices in stack overflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 200–204, 2019. doi: 10.1109/MSR.2019.00040. [p26]

T. Storer. Bridging the chasm: A survey of software engineering practice in scientific programming. *ACM Comput. Surv.*, 50(4), aug 2017. ISSN 0360-0300. doi: 10.1145/3084225. [p26]

M. L. Toro. Understanding the consistency of stack overflow code: A cautionary suggestion. *LC International Journal of STEM (ISSN: 2708-7123)*, 2(1):40–47, 2021. [p26]

A. Zerouali, C. Velázquez-Rodríguez, and C. De Roover. Identifying versions of libraries used in stack overflow code snippets. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 341–345, 2021. doi: 10.1109/MSR52588.2021.00046. [p26]

H. Zhang. *On the Maintenance of Crowdsourced Knowledge on Stack Overflow*. PhD thesis, Queen's University (Canada), 2020. [p26]

T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim. Are code examples on an online q amp;a forum reliable?: A study of api misuse on stack overflow. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 886–896, 2018. doi: 10.1145/3180155.3180260. [p26]

Melina Vidoni
Australian National University, School of Computing
Canberra, Australia
0000-0002-4099-1430
melina.vidoni@anu.edu.au

g2f as a Novel Tool to Find and Fill Gaps in Metabolic Networks

by Daniel Osorio, Kelly Botero, Andrés Pinzón Velasco, Nicolás Mendoza-Mejía, Felipe Rojas-Rodríguez, George Barreto and Janneth González

Abstract During the building of a genome-scale metabolic model, there are several dead-end metabolites and substrates which cannot be imported, produced, nor used by any reaction incorporated in the network. The presence of these dead-end metabolites can block out the net flux of the objective function when it is evaluated through Flux Balance Analysis (FBA), and when it is not blocked, bias in the biological conclusions increase. In this aspect, the refinement to restore the connectivity of the network can be carried out manually or using computational algorithms. The **g2f** package was designed as a tool to find the gaps from dead-end metabolites and fill them from the stoichiometric reactions of a reference, filtering candidate reactions using a weighting function. Additionally, this algorithm allows downloading all the sets of gene-associated stoichiometric reactions for a specific organism from the KEGG database. Our package is compatible with both 4.0.0 and 3.6.0 R versions.

Introduction

Genome-scale metabolic models (GEMs) are multi-compartment metabolic reconstructions that specify the set of chemical reactions catalyzed by an organism (usually hundreds to thousands) covering the metabolic biochemical molecular function of a complete genome (Szappanos et al., 2011). The main goal of these reconstructions is to relate the genome of a given organism with its physiology, incorporating every metabolic transformation that this organism can perform (Agren et al., 2013; Chen et al., 2012). The GEMs are converted into computational models for the simulation of a species-specific metabolism in order to gain insight into the complex interactions that give rise to the metabolic capabilities (Alper et al., 2005; Fong et al., 2005; Cook and Nielsen, 2017). The predictive accuracy of a model depends on the comprehensiveness and biochemical fidelity of the reconstruction (Thiele et al., 2014).

The GEM construction process can be divided into two fundamental stages: (1) The generation of a draft of the reconstructed network. Here, the reactions associated with the enzymes that participate in the metabolism of a particular organism are downloaded from specialized databases such as KEGG, MetaCyc, or ModelSEED (Pham et al., 2019; van Steijn et al., 2019). (2) A refinement of the network is done manually or through the use of computational algorithms (Pham et al., 2019; van Steijn et al., 2019). Similar steps are performed during the construction of a tissue-specific metabolic reconstruction, defined as the subset of reactions included in a genome-scale metabolic reconstruction that are highly associated with the metabolism of a specific tissue (Palsson, 2009; Schultz and Qutub, 2016; van Steijn et al., 2019). These are constructed from the measured gene expression or proteomic data allowing researchers to characterize and predict the metabolic behavior of tissue under any physiological conditions Ataman et al. (2017). It is important to highlight that a drawback of this approach arises from the fact that only the reactions associated with specific enzymes or genes can be mapped from the measured data. Therefore, the spontaneous and non-facilitated-transport reactions are missing in the first stages (Schultz and Qutub, 2016).

If all relevant exchange reactions are available, a high-quality model is expected to be able to carry flux in all its reactions (Agren et al., 2013); thus, a refinement stage in the reconstruction is required to restore the connectivity of the network. In this aspect, the gaps in the draft reconstruction are identified, and candidate reactions to fill the gaps are found using literature and metabolic databases (Satish Kumar et al., 2007; Thiele and Palsson, 2010). The network gaps can be associated with dead-end metabolites, which cannot be imported nor produced by any of the reactions in the network, or metabolites that are not used as substrates or released by any of the reactions. The presence of this type of metabolites can be problematic when the metabolic network is transformed into a steady-state metabolic model; mainly because flux through the network is blocked due to the incomplete connectivity with the rest of the network. Therefore, it is not possible to accurately optimize the metabolic flux distribution under an objective function, increasing the bias in the biological conclusions obtained from the reconstruction (Satish Kumar et al., 2007).

A manual refinement can be performed as an iterative process to assemble a higher confidence compendium of organism-specific metabolic reactions on a draft metabolic reconstruction (Bateman, 2010; Heavner and Price, 2015; Howe et al., 2008). Since the network reconstructions typically involve thousands of metabolic reactions, the model refinement can be a very complex task, which not only requires plenty of time and intensive use of available literature, databases, and experimental data

(Heavner and Price, 2015; Lakshmanan et al., 2014) but also can lead to the introduction of new errors and to overlook old ones (Agren et al., 2013; Machado et al., 2018). These metabolic network gap refinement can also be performed using several algorithms developed for open.source environments, such as Python and GAMS, or in a closed-source environment such as MATLAB (Wang and Marci, 2018). Commonly implemented algorithms are mainly based on optimization procedures to fill the gaps that allow the production of a specific metabolite or give flux for a single biological objective function. Other algorithms modify the directionality of reactions or add new reactions to the model without associated evidence (Table 1)

Algorithm	**Implementation**		**(Open source)**	
	Package	**Environment**	**Package**	**Environment**
"SMILEY"	COBRApy	Python	Yes	Yes
"gapFind" and "gapFill"	-	GAMS	-	Yes
"growMatch"	COBRApy	Python	Yes	Yes
"fastgapfill"	openCOBRA	MATLAB	Yes	No

Table 1: Description and comparison of the methods used for gap find and filling. The available algorithms are presented under the different environments.

Table 1 listed the four most used algorithms for gap filling across three environments. SMILEY, developed by Reed et al. (2006), identifies the minimum number of reactions required to allow the model a specific metabolite production through an optimization function. Reactions to fill the gaps are identified from a universal database of stoichiometric reactions, and the process is carried out one metabolite per time (user-defined). Alternatively, "gapFind" and "gapFill" in GAMS were developed by Satish Kumar et al. (2007) and identified the metabolites ('gapFind') in the metabolic network reconstruction, which cannot be produced under any uptake conditions in both single and multicompartment. Subsequently, 'gapFill' identify the reactions from a customized multi-organism database that restores the connectivity of these metabolites to the original network using optimization-based procedures. In the process, the procedure makes several intra-model modifications such as: (1) modify the directionality of the reactions in the model, (2) add fake external transport mechanisms, and (3) add fake intracellular transport reactions in multicompartment models. "growMatch" was developed by Kumar and Maranas (2009), and it identifies the minimum number of reactions required to allow the model flux to a selected objective function through an optimization algorithm. Reactions to fill the gaps are identified from a universal database of stoichiometric reactions. The process is carried out with one objective function per time (user-defined). Finally, developed by Thiele et al. (2014), the 'fastGapFill' algorithm identifies the blocked reactions through an optimization procedure. It searches candidate reactions to fill the gaps in a universal database of stoichiometric reactions through the 'fastCore' algorithm. This second algorithm computes a compact flux consistent model and uses it to filter and determine the reactions to be added. In the filling process, fake transport reactions between compartments are added.

In this aspect, and with the aim of offering an open-source tool that improves the refinement of drafts network reconstructions and the depuration of metabolic models under the R environment, we introduce the **g2f** R package. This tool includes five functions to identify and fill gaps, calculate the additional cost of a reaction, and depurate metabolic networks of blocked reactions (no activated under any scenario). The implemented *gapFill* algorithm in **g2f** identifies the dead-end metabolites and traces them in a universal database of stoichiometric reactions used as a reference to select candidate reactions to be added. Selected reactions are then filtered by the function *additionCost* considering metabolites present in the original reconstruction to minimize the number of new metabolites to be added. The function calculates the cost of adding a reaction by dividing the amount of non-included metabolites in the reference metabolic network over the total number of metabolites involved in the reaction. The latter is done to minimize the number of false-positive metabolites that could increase the number of new gaps in the model. Also, *blockedReactions* search for blocked reactions, so *gapFill* can fill blocked paths in the network. Finally, *getReactionsList* extracts the reactions from the model in the form of a list of strings, so it can be easily compared with the list of reactions obtained from *getReference*, which downloads specific stoichiometric matrices from KEGG in order to reconstruct specific organism models.

Installation and Functions

The **g2f** package is available for download and installation from the Comprehensive R Archive Networks (CRAN, Hornik (2012)). This package is compatible with R 3.6.0 and 4.0.0 versions. To get the latest stable version of **g2f**, install it directly from GitHub:

Workflow **Input:** A **sybil** metabolic model.
1. with *getReference*: Reference reactions list is retrieved from KEGG database.
2. with *blockedReactions*: Check if there is any dead-end metabolite, the results serve as a guide to the user.
3. with *getReactionsList*: List of reactions is extracted from input metabolic model.
4. with *additionCost*: The addition cost for the reference reactions list can be calculated to do a manual check.
5. with *gapFill*: Find dead-end metabolites and fill the gaps with reactions from the reference list, which are below the addition cost treshold defined. **Loop** user defined times (default = 5) **5.1.** Searches dead-end reactants and products. **5.2.** Calculates the additional cost of the reference reactions. **5.3.** Filters reference reactions with a cost above the threshold. **5.4.** Selects the filtered reactions that have any orphan reactant or product. **5.5.** Fills the gaps in the model with the selected reactions.
Output: List of the added reactions with their additional costs

Table 2: Workflow of **g2f** packet

```
# Install 'devtools' R Package
R> install.packages('devtools')

# Install 'g2f' package
R> setRepositories(ind=1:2)
R> devtools::install_github('gibbslab/g2f')
R> library('g2f')
```

g2f includes 5 functions in order to identify gaps (metabolites not produced or not consumed in any reaction) and fill the gaps from a reference metabolic reconstruction. Briefly, the gap-filling reconstruction is based on the stoichiometric reaction matrix either from a specific model or by the complete set of gene-associated stoichiometric reactions for a specific organism from the KEGG database using a weighting function. Table 3 summarizes the functions contained in the **g2f** R package.

Function	**Description**
blockedReactions	Identifies blocked reactions in a metabolic network.
additionCost	Calculates the cost of addition of a stoichiometric reaction.
getReactionsList	Extract the reaction list from a model.
getReference	Download all stoichiometric reactions from the KEGG database.
gapFill	Find and fill gaps in a metabolic network.

Table 3: Descriptions of **g2f** available functions.

Downloading reference data from KEGG database

The KEGG database is a resource, widely used as a reference in genomics, metagenomics, metabolomics, and other studies. Moreover, KEGG has been used for modeling and simulation in systems biology, specifically in GEMs (Kanehisa, 2006; Kanehisa et al., 2016; Martín-Jiménez et al., 2017). Currently, the database includes complete genomes, biological pathways, and the associated stoichiometric reactions for 542 eukaryotes, 5979 bacteria, and 334 archaea. The **g2f**'s `getReference` function downloads all the gene-associated KeggOrthology (KO) stoichiometric reactions from KEGG and their correspondent E.C. numbers for a customized organism, through the use of KEGG organism ID. Based on the KOs associated with the reactions, their respective gene-protein-reaction is constructed as follows: all genes associated with a given KO are linked by an AND operator. After that, when a reaction has more than one associated KO, previously linked genes are now joined by an OR operator. As an example, to download all the stoichiometric reactions (1492) associated with *Escherichia coli*, just type:

```
R> e.coli <- getReference(organism = "eco")
```

Identify blocked reactions

To identify the blocked reactions included in a metabolic model, the `blockedReactions` function sets each one of the reactions included in the model (one at the time) as the objective function and optimizes the system through Flux Balance Analysis (FBA). Reactions that are not participating in any possible solution during all evaluations are returned as a blocked reaction.

As an example, we identify the blocked reactions in the E. coli core metabolic model included in the **sybil** package (Gelius-Dietrich et al., 2013).

```
R> data("Ec_core")
R> blockedReactions(Ec_core)

|=============================================================| 100%
[1]  "EX_fru(e)" "EX_fum(e)"  "EX_mal_L(e)"  "FUMt2_2"  "MALt2_2"
```

Calculating the additional cost

Adding new reactions in order to fill gaps can be an easy path to increase the number of dead-end metabolites (Hosseini and Marashi, 2017). Therefore, as a strategy to reduce the possible addition of new dead-end metabolites into the system, the `additionCost` function calculates the cost of adding new metabolites based on metabolites that constitute the new reaction and those that compose the stoichiometric reactions already present in the metabolic reconstruction (Equation 1). Values of the function represent a weight ranging between 0 and 1.

$$additionCost = \frac{n(metabolites(newReaction)) \notin (metabolites(reactionList))}{n(metabolites(newReaction)} \tag{1}$$

As an example, we select a sample of reactions from the downloaded reference for E. coli and calculate the additional cost for the remaining reactions (6 first values are shown).

```
R> reactionList <- sample(e.coli$reaction,10)
R> head(
    +  additionCost(reaction = e.coli$reaction,
    +  reference = reactionList)
    +  )
[1] 1.0000000 1.0000000 1.0000000 0.8000000 0.8333333 1.0000000
```

To understand the results of the `additionCost`, we present two examples for the glutamine synthetase reaction in the glutamate metabolism of E. coli core model.

```
[c]: ATP + Glu-L + Nh4 --> ADP + Gln-L + h + pi
```

The reaction takes as input Adenosine triphosphate (ATP), L-Glutamate (Glu-L), and Ammonium (Nh4) and produces Adenosine diphosphate (ADP), L-Glutamine (Gln-L), H+ (h), and inorganic Phosphate (pi) in the cytoplasm. We are going to assume that this reaction is going to be added to the model and that the number of metabolites to be added change between two conditions. In the first case, the reaction would be evaluated by `additionCost`, but one of the seven metabolites is not present on the list of reactions of the complete model. In the second situation, four of the seven metabolites are not present in the metabolite list of the model. By dividing the number of metabolites to be added by the total number of metabolites in the reaction, `additionCost` produces 0.14 and 0.57 as resulting values for the two conditions respectively. In this sense, if we pick a threshold of 0.2 for the `gapfill` the first case would allow the reaction to be added but not the second condition. By using a threshold of 0.2 is possible to set a medium point for the reaction addition. Where higher values are more permissive and lower values are more restrictive.

"Gap find and fill" performing, input and syntaxis

To identify network gaps in a metabolic model and fill them from a reference network, the `gapFill` function performs several steps: (1) The dead-end metabolites are identified from the stoichiometric matrix, (2) the candidate reactions are to be added by comparing the metabolites against the metabolite list of the model, (3) the additional cost of each candidate reaction is calculated, (4) the candidate reactions with an additional cost lower or equal to the user-defined limit are added to the reaction list. Finally, the process returns to step 1 until no more original-gaps can be filled under the user-defined limit. The function returns a set of candidate stoichiometric reactions to fill the original-gaps included in the metabolic network.

As an example, we show how to fill dead-end metabolites included in the previously selected sample using all downloaded stoichiometric reactions from the KEGG database for E. coli as the reference.

```
R> reactionsAdded <- gapFill(reactionList = reactionList,
          +      reference = e.coli$reaction,
          +      limit = 1/4
          +      )
48% gaps filled in the last iteration
26% gaps filled in the last iteration
13% gaps filled in the last iteration
13% gaps filled in the last iteration
4% gaps filled in the last iteration

R> head(reactionsAdded)
addCost                                                                  react
1    0.00      L-Glutamine + D-Fructose 6-phosphate <=> L-Glutamate + D-Glucosamine
          6-phosphate
2    0.25                            ATP + Pyruvate <=> ADP + Phosphoenolpyruvate
3    0.00                                                     ATP + AMP <=> 2 ADP
4    0.25                                               ATP + dTDP <=> ADP + dTTP
5    0.00  ATP + 5-Fluorouridine diphosphate <=> ADP + 5-Fluorouridine triphosphate
6    0.25                                                 ATP + UDP <=> ADP + UTP
```

The output is a data frame with the reactions that were found to fill the gaps in the model, with the corresponding `additionCost` calculated for each one.

Compatibility

In order to provide compatibility, **g2f** implements `getReactionsList` a function that helps to extract the reactions of a **sybil** model as a list of strings, each string being a reaction, which is the input format of `gapFill` accepts.

In the examples before, we used a reduced version for the reference organism of E.coli from KEGG. Now we will use a converted model to SBML using KEGG2SBML (Moutselos et al., 2009) from (Akiya Jouraku and Kitano, 2008), which will be converted into **sybil** with the help of the **sybilSBLM** package, and then the reactions list will be extracted to use them with the `gapFill` function. Note that we have done this because the name of the reaction metabolites in the model should be the same as the ones used in KEGG, and the E.coli core metabolic model included in the **sybil** package does not meet this requirement.

```
# Install and import sybilSBML package
R> install.packages('sybilSBML')
R> library('sybilSBML')

# Read the SBML and convert it to sybil
R> mod <- readSBMLmod("eco/eco00730.xml", bndCond = FALSE)

# Extract the model's reactions
R> react <- getReactionsList(mod)

# Fill the gaps
R> reactionsAdded <- gapFill(reactionList = react$react,
          reference = e.coli$reaction,
          limit = 1/4
)

20% gaps filled in the last iteration
0% gaps filled in the last iteration
0% gaps filled in the last iteration
0% gaps filled in the last iteration
0% gaps filled in the last iteration
```

```
  addCost                               react
1       0            ATP + ADP <=> ADP + ATP
2       0 ATP + H2O <=> ADP + Orthophosphate
```

g2f performance

We tested the performance of **g2f** against the most used platforms for gap-filling in the metabolic networks using a computer with i7 8750h 2.2GHz processor and 12Gb DDR4 Ram. We compared the performance of R package **g2f**, Python **cobrapy** `gapfill` function, and Matlab **COBRA** `fastgapfilling` function (Table 4). The benchmark was performed for each gap-filling algorithm by deleting 10 random reactions across the E. coli core model (Orth et al., 2010).

Platform	Limit	TicToc (sec)	Solution
R: g2f – "gapfill"	0.1	2.83	Feasible
	0.15	2.76	
	0.2	2.73	
	0.25	6.91	
Python: Cobrapy – "gapfill"	-	1.369	Unfeasible
Matlab: COBRA – "fastgapfill" [Cplex solver]	0.1	7.858	Feasible
	0.15	8.836	
	0.2	9.001	
	0.25	5.695	

Table 4: Performance of **g2f** compared with other gap-filling algorithms. The limit is associated with the threshold for the limit of gap-filling. TicToc was the methodological approach used to measure the performance time. The solution is the capacity of the model to run a FBA after the gap fill function was run. A single iteration of the gap-filling algorithm Cobrapy-"gapfill" was unable to generate a suitable FBA.

Considering the computational performance and flux recovery across the network (FBA solution), **g2f** arises as a suitable method for Genome-scale metabolic network reconstructions gap filling using curated models as reference.

Application

A wide variety of open-source, paid software, and webtools have been developed to fill the gaps in automated or manual metabolic reconstructions (Karp et al., 2018; Machado et al., 2018; Prigent et al., 2017). Performing a gap-filling accurately is a challenging task considering the possibility of overestimating reaction addition or excluding metabolites from the filling by inquorate thresholds (Pan and Reed, 2018). **g2f** offers an R based open-source alternative capable of integrating with systems biology packages such as **sybil** (Gelius-Dietrich et al., 2013) or **minVal** (Osorio et al., 2017) as well as big projects such as Recon3D (Brunk et al., 2018) or the Human Metabolic Atlas (Pornputtapong et al., 2015). Finally, considering that the majority of metabolic models are derived from annotated genomes where not all the enzymes are known, **g2f** offers the possibility to optimize the topology of public available metabolic models or automated metabolic reconstructions.

Conclusions

We developed **g2f**, a novel R package to, find dead-end metabolites in a genome-scale metabolic reconstruction and fill the reaction gaps with metabolites available in a stoichiometric matrix from a reference model. Additionally, **g2f** filters the candidate reactions using a weighting function and a user-defined limit. We depicted the functions included in the package using the E. coli reference model downloaded from the KEGG database, and the core metabolic model included in the **sybil** package. Finally, the performance of **g2f** was compared with other gap-filling algorithms (**Cobrapy** – `gapfill` and Matlab:**COBRA** – `fastgapfill`), showing an adequate feasibility and performance speed.

Summary

Dead-end metabolites are a major drawback in genome-scale metabolic reconstruction and analysis. Since there is a lack of available tools to solve this situation in the R environment, hereby, we introduce the **g2f** package to find and fill dead-end metabolites in a given reconstruction based on a reference template. Our method allows users to filter candidate reactions using a weighting function and a user-defined limit. We show step by step the functionality of each procedure included in the package using a reference model downloaded from the KEGG database for Escherichia coli and the core metabolic model included in the **sybil** package.

Acknowledgements

This work was supported by the Pontificia Universidad Javeriana, Bogotá, Colombia, and Minciencias IDs 7740, 8845, and 20304 to JG. We thank the anonymous reviewers and testers for their helpful comments and suggestions to improve the CRAN package.

Bibliography

R. Agren, L. Liu, S. Shoaie, W. Vongsangnak, I. Nookaew, and J. Nielsen. The RAVEN Toolbox and Its Use for Generating a Genome-scale Metabolic Model for Penicillium chrysogenum. *PLoS Computational Biology*, 9(3):e1002980, Mar. 2013. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002980. URL https://dx.plos.org/10.1371/journal.pcbi.1002980. [p28, 29]

A. F. Akiya Jouraku, Nobuyuki Ohta and H. Kitano. systems-biology, 2008. URL http://www.systems-biology.org/001/001.html. [p32]

H. Alper, Y.-S. Jin, J. Moxley, and G. Stephanopoulos. Identifying gene targets for the metabolic engineering of lycopene biosynthesis in Escherichia coli. *Metabolic Engineering*, 7(3):155–164, May 2005. ISSN 10967176. doi: 10.1016/j.ymben.2004.12.003. URL https://linkinghub.elsevier.com/retrieve/pii/S1096717604000849. [p28]

M. Ataman, D. F. Hernandez Gardiol, G. Fengos, and V. Hatzimanikatis. redGEM: Systematic reduction and analysis of genome-scale metabolic reconstructions for development of consistent core metabolic models. *PLoS Computational Biology*, 13(7), 2017. ISSN 1553-734X. doi: 10.1371/journal.pcbi.1005444. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5519011/. [p28]

A. Bateman. Curators of the world unite: the International Society of Biocuration. *Bioinformatics*, 26(8):991–991, Apr. 2010. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btq101. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btq101. [p28]

E. Brunk, S. Sahoo, D. C. Zielinski, A. Altunkaya, A. Dräger, N. Mih, F. Gatto, A. Nilsson, G. A. Preciat Gonzalez, M. K. Aurich, A. Prlić, A. Sastry, A. D. Danielsdottir, A. Heinken, A. Noronha, P. W. Rose, S. K. Burley, R. M. T. Fleming, J. Nielsen, I. Thiele, and B. O. Palsson. Recon3d enables a three-dimensional view of gene variation in human metabolism. *Nature Biotechnology*, 36(3): 272–281, Mar. 2018. ISSN 1087-0156, 1546-1696. doi: 10.1038/nbt.4072. URL http://www.nature.com/articles/nbt.4072. [p33]

N. Chen, I. J. d. Val, S. Kyriakopoulos, K. M. Polizzi, and C. Kontoravdi. Metabolic network reconstruction: advances in in silico interpretation of analytical information. *Current Opinion in Biotechnology*, 23(1):77–82, Feb. 2012. ISSN 09581669. doi: 10.1016/j.copbio.2011.10.015. URL https://linkinghub.elsevier.com/retrieve/pii/S0958166911007129. [p28]

D. J. Cook and J. Nielsen. Genome-scale metabolic models applied to human health and disease. *WIREs Systems Biology and Medicine*, 9(6):e1393, 2017. ISSN 1939-005X. doi: 10.1002/wsbm.1393. URL http://onlinelibrary.wiley.com/doi/abs/10.1002/wsbm.1393. [p28]

S. S. Fong, A. P. Burgard, C. D. Herring, E. M. Knight, F. R. Blattner, C. D. Maranas, and B. O. Palsson. In silico design and adaptive evolution of *Escherichia coli* for production of lactic acid. *Biotechnology and Bioengineering*, 91(5):643–648, Sept. 2005. ISSN 00063592, 10970290. doi: 10.1002/bit.20542. URL http://doi.wiley.com/10.1002/bit.20542. [p28]

G. Gelius-Dietrich, A. Desouki, C. Fritzemeier, and M. J. Lercher. sybil – Efficient constraint-based modelling in R. *BMC Systems Biology*, 7(1):125, 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-125. URL http://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-7-125. [p31, 33]

B. D. Heavner and N. D. Price. Transparency in metabolic network reconstruction enables scalable biological discovery. *Current Opinion in Biotechnology*, 34:105–109, Aug. 2015. ISSN 09581669. doi: 10.1016/j.copbio.2014.12.010. URL `https://linkinghub.elsevier.com/retrieve/pii/S0958166914002250`. [p28, 29]

K. Hornik. The Comprehensive R Archive Network: The Comprehensive R Archive Network. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(4):394–398, July 2012. ISSN 19395108. doi: 10.1002/wics.1212. URL `http://doi.wiley.com/10.1002/wics.1212`. [p29]

Z. Hosseini and S.-A. Marashi. Discovering missing reactions of metabolic networks by using gene co-expression data. *Scientific Reports*, 7(1):41774, Mar. 2017. ISSN 2045-2322. doi: 10.1038/srep41774. URL `http://www.nature.com/articles/srep41774`. [p31]

D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre, S. Twigger, O. White, and S. Yon Rhee. The future of biocuration. *Nature*, 455(7209):47–50, 2008. ISSN 1476-4687. doi: 10.1038/455047a. URL `https://www.nature.com/articles/455047a`. [p28]

M. Kanehisa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34(90001):D354–D357, Jan. 2006. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gkj102. URL `https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkj102`. [p30]

M. Kanehisa, Y. Sato, M. Kawashima, and M. Furumichi. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research*, 44(D1):D457–D462, Jan. 2016. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gkv1070. URL `https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv1070`. [p30]

P. D. Karp, D. Weaver, and M. Latendresse. How accurate is automated gap filling of metabolic models? *BMC Systems Biology*, 12(1):73, Dec. 2018. ISSN 1752-0509. doi: 10.1186/s12918-018-0593-7. URL `https://bmcsystbiol.biomedcentral.com/articles/10.1186/s12918-018-0593-7`. [p33]

V. S. Kumar and C. D. Maranas. GrowMatch: An Automated Method for Reconciling In Silico/In Vivo Growth Predictions. *PLoS Computational Biology*, 5(3):e1000308, Mar. 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000308. URL `https://dx.plos.org/10.1371/journal.pcbi.1000308`. [p29]

M. Lakshmanan, G. Koh, B. K. S. Chung, and D.-Y. Lee. Software applications for flux balance analysis. *Briefings in Bioinformatics*, 15(1):108–122, Jan. 2014. ISSN 1467-5463, 1477-4054. doi: 10.1093/bib/bbs069. URL `https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbs069`. [p29]

D. Machado, S. Andrejev, M. Tramontano, and K. R. Patil. Fast automated reconstruction of genome-scale metabolic models for microbial species and communities. *Nucleic Acids Research*, 46(15): 7542–7553, Sept. 2018. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gky537. URL `https://academic.oup.com/nar/article/46/15/7542/5042022`. [p29, 33]

C. A. Martín-Jiménez, D. Salazar-Barreto, G. E. Barreto, and J. González. Genome-Scale Reconstruction of the Human Astrocyte Metabolic Network. *Frontiers in Aging Neuroscience*, 9, Feb. 2017. ISSN 1663-4365. doi: 10.3389/fnagi.2017.00023. URL `http://journal.frontiersin.org/article/10.3389/fnagi.2017.00023/full`. [p30]

K. Moutselos, I. Kanaris, A. Chatziioannou, I. Maglogiannis, and F. N. Kolisis. KEGGconverter: a tool for the in-silico modelling of metabolic networks of the KEGG pathways database. *BMC Bioinformatics*, 10(1):324, 2009. ISSN 1471-2105. doi: 10.1186/1471-2105-10-324. URL `https://doi.org/10.1186/1471-2105-10-324`. [p32]

J. D. Orth, B. O. Palsson, and R. M. T. Fleming. Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide. *EcoSal Plus*, 4(1), Sept. 2010. ISSN 2324-6200. doi: 10.1128/ecosalplus.10.2.1. URL `http://www.asmscience.org/content/journal/ecosalplus/10.1128/ecosalplus.10.2.1`. [p33]

D. Osorio, J. González, and A. Pinzón. minval: An R package for MINimal VALidation of Stoichiometric Reactions. *The R Journal*, 9(1):114, 2017. ISSN 2073-4859. doi: 10.32614/RJ-2017-031. URL `https://journal.r-project.org/archive/2017/RJ-2017-031/index.html`. [p33]

B. Palsson. Metabolic systems biology. *FEBS Letters*, 583(24):3900–3904, Dec. 2009. ISSN 00145793. doi: 10.1016/j.febslet.2009.09.031. URL `http://doi.wiley.com/10.1016/j.febslet.2009.09.031`. [p28]

S. Pan and J. L. Reed. Advances in gap-filling genome-scale metabolic models and model-driven experiments lead to novel metabolic discoveries. *Current Opinion in Biotechnology*, 51:103–108, June 2018. ISSN 09581669. doi: 10.1016/j.copbio.2017.12.012. URL https://linkinghub.elsevier.com/retrieve/pii/S0958166917302045. [p33]

N. Pham, R. van Heck, J. van Dam, P. Schaap, E. Saccenti, and M. Suarez-Diez. Consistency, Inconsistency, and Ambiguity of Metabolite Names in Biochemical Databases Used for Genome-Scale Metabolic Modelling. *Metabolites*, 9(2):28, Feb. 2019. ISSN 2218-1989. doi: 10.3390/metabo9020028. URL http://www.mdpi.com/2218-1989/9/2/28. [p28]

N. Pornputtapong, I. Nookaew, and J. Nielsen. Human metabolic atlas: an online resource for human metabolism. *Database*, 2015, Jan. 2015. ISSN 1758-0463. doi: 10.1093/database/bav068. URL https://academic.oup.com/database/article/doi/10.1093/database/bav068/2433201. [p33]

S. Prigent, C. Frioux, S. M. Dittami, S. Thiele, A. Larhlimi, G. Collet, F. Gutknecht, J. Got, D. Eveillard, J. Bourdon, F. Plewniak, T. Tonon, and A. Siegel. Meneco, a Topology-Based Gap-Filling Tool Applicable to Degraded Genome-Wide Metabolic Networks. *PLOS Computational Biology*, 13(1): e1005276, Jan. 2017. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005276. URL https://dx.plos.org/10.1371/journal.pcbi.1005276. [p33]

J. L. Reed, T. R. Patel, K. H. Chen, A. R. Joyce, M. K. Applebee, C. D. Herring, O. T. Bui, E. M. Knight, S. S. Fong, and B. O. Palsson. Systems approach to refining genome annotation. *Proceedings of the National Academy of Sciences*, 103(46):17480–17484, Nov. 2006. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0603364103. URL http://www.pnas.org/cgi/doi/10.1073/pnas.0603364103. [p29]

V. Satish Kumar, M. S. Dasika, and C. D. Maranas. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8(1):212, 2007. ISSN 14712105. doi: 10.1186/1471-2105-8-212. URL http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-212. [p28, 29]

A. Schultz and A. A. Qutub. Reconstruction of Tissue-Specific Metabolic Networks Using CORDA. *PLOS Computational Biology*, 12(3):e1004808, Mar. 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004808. URL http://dx.plos.org/10.1371/journal.pcbi.1004808. [p28]

B. Szappanos, K. Kovács, B. Szamecz, F. Honti, M. Costanzo, A. Baryshnikova, G. Gelius-Dietrich, M. J. Lercher, M. Jelasity, C. L. Myers, B. J. Andrews, C. Boone, S. G. Oliver, C. Pál, and B. Papp. An integrated approach to characterize genetic interaction networks in yeast metabolism. *Nature Genetics*, 43(7):656–662, July 2011. ISSN 1061-4036, 1546-1718. doi: 10.1038/ng.846. URL http://www.nature.com/articles/ng.846. [p28]

I. Thiele and B. O. Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature Protocols*, 5(1):93–121, Jan. 2010. ISSN 1754-2189, 1750-2799. doi: 10.1038/nprot.2009.203. URL http://www.nature.com/articles/nprot.2009.203. [p28]

I. Thiele, N. Vlassis, and R. M. T. Fleming. fastGapFill: efficient gap filling in metabolic networks. *Bioinformatics*, 30(17):2529–2531, Sept. 2014. ISSN 1367-4803, 1460-2059. doi: 10.1093/bioinformatics/btu321. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu321. [p28, 29]

L. van Steijn, F. J. Verbeek, H. P. Spaink, and R. M. Merks. Predicting metabolism from gene expression in an improved whole-genome metabolic network model of danio rerio. *Zebrafish*, 16(4):348–362, 2019. ISSN 1545-8547. doi: 10.1089/zeb.2018.1712. URL https://www.liebertpub.com/doi/10.1089/zeb.2018.1712. Publisher: Mary Ann Liebert, Inc., publishers. [p28]

H. Wang and S. Marci. RAVEN 2.0: A versatile toolbox for metabolic network reconstruction and a case study on Streptomyces coelicolor. *PLOS Computational Biology*, page 17, 2018. [p29]

Daniel Osorio
Grupo de Investigación en Bioinformática y Biología de Sistemas. Instituto de Genética, Universidad Nacional de Colombia.
Bogotá
Colombia
ORCiD: 0000-0003-4424-8422
dcosorioh@unal.edu.co

Kelly Botero
Grupo de Investigación en Bioinformática y Biología de Sistemas. Instituto de Genética, Universidad Nacional

de Colombia.
Bogotá
Colombia
kjboteroo@unal.edu.co

Andrés Pinzón Velasco
Grupo de Investigación en Bioinformática y Biología de Sistemas. Instituto de Genética, Universidad Nacional de Colombia.
Bogotá
Colombia
ampinzonv@unal.edu.co

Nicolás Mendoza-Mejía
Grupo de Investigación en Bioquímica Experimental y Computacional. Departamento de Nutrición y Bioquímica, Facultad de Ciencias, Pontificia Universidad Javeriana.
Bogotá
Colombia
mendozamejian@gmail.com

Felipe Rojas-Rodriguez
Division of Molecular Pathology, The Netherlands Cancer Institute - Antoni van Leeuwenhoek Hospital.
Amsterdam 1066 CX
The Netherlands.
f.rojas@nki.nl

George E. Barreto
Department of Biological Sciences, University of Limerick.
V94 T9PX Limerick
Ireland
george.barreto@ul.ie

Janneth González
Grupo de Investigación en Bioquímica Experimental y Computacional. Departamento de Nutrición y Bioquímica, Facultad de Ciencias, Pontificia Universidad Javeriana.
Bogotá
Colombia
ORCiD: 0000-0003-2009-3374
janneth.gonzalez@javeriana.edu.co

lg: An R package for Local Gaussian Approximations

by Håkon Otneim

Abstract The package **lg** for the R programming language provides implementations of recent methodological advances on applications of the local Gaussian correlation. This includes the estimation of the local Gaussian correlation itself, multivariate density estimation, conditional density estimation, various tests for independence and conditional independence, as well as a graphical module for creating dependence maps. This paper describes the **lg** package, its principles, and its practical use.

Introduction

Tjøstheim and Hufthammer (2013) propose the *local Gaussian correlation* (LGC) as a new measure of statistical dependence between two stochastic variables X_1 and X_2, which has the following important property yet unrivaled in the literature: It can separate between positive and negative nonlinear dependence while still reducing to the ordinary Pearson correlation coefficient if X_1 and X_2 are jointly normally distributed. The R-package **localgauss** (Berentsen et al., 2014) provides two important functions in this context; one that calculates the sample LGC based on observed values of (X_1, X_2), and one that uses the estimated LGC to perform a local test of independence between X_1 and X_2 as described in detail by Berentsen and Tjøstheim (2014).

We have lately seen a number of new applications of the LGC that the **localgauss** package does not support, however. Støve et al. (2014) use the LGC to test for financial contagion across markets during crises. Otneim and Tjøstheim (2017) present a procedure for estimating multivariate density functions via the LGC, which Otneim and Tjøstheim (2018) modify in order to compute estimates of conditional density functions. Lacal and Tjøstheim (2017) present a test for serial independence within a time series, which Lacal and Tjøstheim (2018) extend in order to include a test for cross-correlation between two time series. Finally, Otneim and Tjøstheim (2021) develop the local Gaussian *partial* correlation (LGPC) as a measure of conditional dependence and a corresponding test for conditional independence.

This paper describes the **lg** package (Otneim, 2019), which provides a unified framework to implement all these methods, as well as a tool for visualizing the LGC and LGPC as dependence maps. Jordanger and Tjøstheim (2020) use the LGC in spectral analysis of time series, but those methods have their own computational ecosystem in the **localgaussSpec** package (Jordanger, 2018).

In Section 2.2, we provide a brief introduction to the LGC as well as the methods and applications referred to above. In Section 2.3, we describe the core function in the **lg** package and move on to demonstrate the implementation of various applications in Section 2.4. We conclude this paper in Section 2.5 by demonstrating the graphical capabilities of the **lg** package.

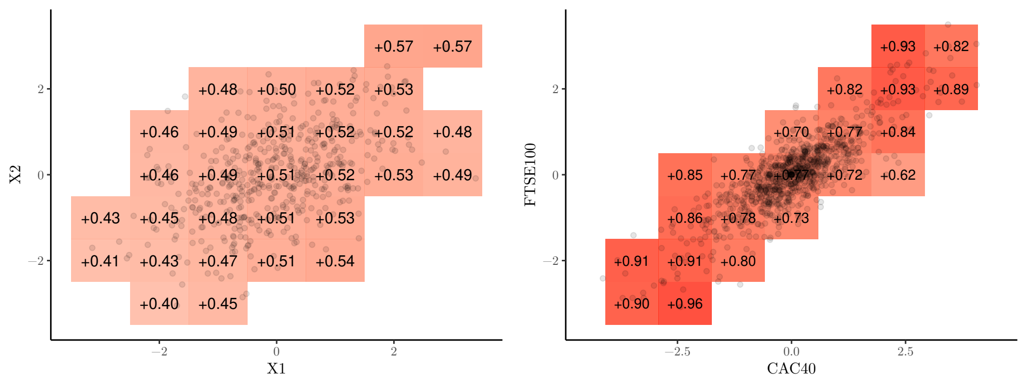
Statistical background

Consider a random vector X having the unknown probability density function $f_X(x)$. It is a standard task to estimate f_X based on a random sample $X_1, \ldots, X_n$, and the statistical literature provides an abundance of methods to accomplish this. One may, for example, make the assumption that the unknown density function has a particular parametric form, $f_X \in F_\theta$, where $F_\theta = \{f(x;\theta), \theta \in \Theta\}$ is a family of probability density functions indexed by some parameter θ, and where Θ is the parameter space. Under this assumption, we will typically produce an estimate of the parameter θ, written $\widehat{\theta}$, using the maximum likelihood method. The estimated probability density function is then given as $\widehat{f}_X(x) = f\left(x;\widehat{\theta}\right)$.

A different approach is to estimate $f_X(\cdot)$ without any prior parametric assumptions. The classical method for nonparametric density estimation is the kernel estimator

$$\widehat{f}_X(x) = \frac{1}{nb}\sum_{i=1}^{n} K\left(\frac{X_i - x}{b}\right),$$

where K is a symmetric density function (the kernel) and b is a tuning parameter (the bandwidth) that controls the smoothness of the estimate $\widehat{f}_X(\cdot)$. See Silverman (1986) for an introduction to this topic. There is also a massive statistical literature on density estimation containing extensions and



(a) Estimated local Gaussian correlations between two jointly normal variables having correlation equal to 0.5, based on 1000 observations.

(b) Local Gaussian correlation between daily returns from the CAC40 and FTSE100 stock indices based on 1000 consecutive trading days.

Figure 1: Two dependence maps

improvements to the classical methods to be used in various practical situations.

Hjort and Jones (1996) provide one such idea. They consider a parametric family F_θ, but instead of searching for a single parameter value θ_0 for which $f_X(x) = f(x;\theta_0)$ (or approximately so), they rather assert that different members of F_θ may approximate f_X locally in different parts of its domain. In other words, they seek to estimate a parameter *function* $\theta_0(x)$ for which $f_X(x) = f(x;\theta_0(x))$ (or approximately so), and do this by maximizing a *local likelihood function* in each point x:

$$\begin{aligned}\widehat{\theta}(x) &= \arg\max_{\theta\in\Theta} L_n(\theta, x) \\ &= \arg\max_{\theta\in\Theta} \frac{1}{nb}\sum_{i=1}^{n} K\left(\frac{X_i - x}{b}\right)\log f(X_i;\theta) - \int \frac{1}{b}K\left(\frac{y-x}{b}\right) f(y;\theta)\,\mathrm{d}y, \qquad (1)\end{aligned}$$

where, again, K is a symmetric density function and b is a bandwidth parameter that controls the smoothness of the estimate. The second term in the local likelihood function is a penalty that ensures that the estimated density $\widehat{f}_X(x) = f\left(x;\widehat{\theta}(x)\right)$ converges correctly to the true density function $f_X(x)$ as the sample size n increases to infinity and the bandwidth b decreases towards zero. See Hjort and Jones (1996) for a detailed discussion about this construction.

Tjøstheim and Hufthammer (2013) consider the bivariate case $X = (X_1, X_2)$ and take F_θ to be the family of bivariate normal distributions consisting of densities on the form

$$\begin{aligned}f(x;\theta) &= \psi(x_1, x_2; \mu_1, \mu_2, \sigma_1, \sigma_2, \rho) \\ &= \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \\ &\quad\times \exp\left\{-\frac{1}{2(1-\rho^2)}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2} - 2\rho\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2}\right)\right\}, \qquad (2)\end{aligned}$$

where $\theta = (\mu_1, \mu_2, \sigma_1, \sigma_2, \rho)$ is the vector of parameters. Using a sample $\{X_{1i}, X_{2i}\}$, $i = 1, \ldots, n$, they estimate θ locally in the point x by maximizing the local likelihood function (1), producing

$$\widehat{\theta}(x) = \left(\widehat{\mu}_1(x), \widehat{\mu}_2(x), \widehat{\sigma}_1(x), \widehat{\sigma}_2(x), \widehat{\rho}(x)\right),$$

and take special interest in the estimated correlation function $\widehat{\rho}(x)$ (i.e., the LGC) because it serves as an attractive local measure of statistical dependence between X_1 and X_2. They show that the LGC reveals many types of nonlinear statistical dependence that are not captured by the ordinary (global) Pearson correlation coefficient. Furthermore, the LGC distinguishes between positive and negative dependence and reduces to the Pearson ρ if X_1 and X_2 are jointly normal. We refer to Tjøstheim and Hufthammer (2013) for a detailed treatment of the theoretical foundations of the LGC as well as several examples and rather present two simple illustrations at this point in order to demonstrate the concept.

In Figure 1, we have plotted the estimated LGC for two bivariate data sets on a grid; 1000 simulated observations from a binormal distribution having correlation equal to 0.5, and the daily return on the CAC40 and FTSE100 stock indices on 1000 consecutive trading days starting on May 5th 2014

(Datastream, 2018). In the first panel, we see that the estimated local correlation coincides with the global correlation, except for the estimation error which is comparable to the uncertainty observed in other nonparametric estimation methods such as the kernel density estimator (see, for instance, Otneim and Tjøstheim (2017) for a formal asymptotic analysis of relevant convergence rates). In the second panel, we see clearly that the local correlation, and thus the dependence, is stronger in the lower left and upper right regions of the distribution than in the central parts. The phenomenon of local dependence is well known in the financial literature, and using the LGC it can be measured, interpreted, and visualized in a natural way. The interpretation of this particular figure is that extreme observations on the two stock indices are more strongly dependent than the less extreme observations.

One may obtain these particular estimates from the older **localgauss** package (as well as the **lg** package, of course), but the plotting routine that was used to produce these figures is included in the **lg** package and will be described in more detail in Section 2.5.

Taking the LGC as a measure of dependence opens up a number of possibilities to construct statistical tests. Berentsen and Tjøstheim (2014) show that $\rho(x) \equiv 0$ implies that X_1 and X_2 are independent. They show further that independence between X_1 and X_2 implies $\rho(x) \equiv 0$ if the population values of the local mean and standard deviation functions satisfy the following conditions: $\mu_i(x_1, x_2) = \mu_i(x_i)$ and $\sigma_i(x_1, x_2) = \sigma_i(x_i)$ for $i = 1, 2$. Equivalence between independence and $\rho(x) \equiv 0$ holds in general if the observations have been suitably transformed according to a procedure presented later in this section. It follows then that departures from $\widehat{\rho}(x) \equiv 0$ may be taken as evidence against the hypothesis that X_1 and X_2 are statistically independent. Berentsen and Tjøstheim (2014) formalize this notion by testing whether $\rho(x) \equiv 0$ for all $x \in S \subset \mathbb{R}^2$ using the test statistic

$$T_{n,b} = \int_S h(\widehat{\rho}(x))\, \mathrm{d}F_n(x) \tag{3}$$

for some non-negative function h, for example $h(x) = x^2$ or $h(x) = |x|$. Critical values may be obtained by permutations of the data under the null hypothesis, and we demonstrate the implementation of this test using the **lg** package in Section 2.4.2.

Consider next the stationary time series $\{X_t\}$. The autocorrelation function (ACF) $\rho_k = \rho(X_t, X_{t-k})$ is a well known concept for describing the serial dependence in the time series, but the ACF is, again, only capable to completely capture *linear* serial dependence. Lacal and Tjøstheim (2017) seek to remedy this by rather calculating the local correlation between X_t and X_{t-k}. This leads to a test for serial independence in a natural way. In fact, this work is mainly a theoretical exercise in order to accommodate time series dependence. Testing for independence between X_t and X_{t-k} using observations $\{X_t, X_{t-k}\}_{t=k+1}^{T}$ leads to the same test statistic (3) and bootstrap procedure as the test for independence between X_1 and X_2 that we described above.

Lacal and Tjøstheim (2018) extend this problem to test for serial cross-dependence between two time series $\{X_t, Y_t\}$ by measuring the LGC between X_t and Y_{t-k}. Departures from $\widehat{\rho}(x, y) \equiv 0$ are, again, taken as evidence against independence, and the test statistic (3) provides an aggregate measure of this discrepancy in the specified region S. In this case, however, we can not obtain replicates of the test statistic under the null hypothesis by simple permutations of the data. Lacal and Tjøstheim (2018) suggest two block bootstrap procedures instead to this end, using fixed and random block sizes, respectively. The tests for serial dependence and serial cross-dependence are both implemented in the **lg** package, as we demonstrate in Section 2.4.2.

We find another application of the local Gaussian approximation in work by Støve et al. (2014), who measure and test for financial contagion. They define contagion as "a significant increase in cross-market linkages after a shock to one country" (Forbes and Rigobon, 2002, p. 2223) and employ the LGC to quantify this potential linkage. The authors estimate the LGC on a grid $\{x_1, x_2\}_k$, $k = 1, \ldots, K$ along the diagonal $D = \{(x_1, x_2) : x_1 = x_2\}$ *before* and *after* some critical event in the financial markets, denoted as the crisis (C) and the non-crisis (NC) periods, respectively. They compare the two estimates using the following test statistic,

$$T_{n,b}^{D} = \sum_{k=1}^{K} \{\widehat{\rho}_C(x_k, x_k) - \widehat{\rho}_{NC}(x_k, x_k)\}\, w(x_k, x_k),$$

where $w(\cdot, \cdot)$ is a weight function that serve the same purpose as the integration area S in (3). In this case, Støve et al. (2014) show that a standard bootstrap will suffice in order to produce approximate replicates of $T_{n,b}^{D}$ under the null hypothesis of no financial contagion, and we demonstrate the implementation of this test using the **lg** package in Section 2.4.3.

Although the original work by Hjort and Jones (1996) provide a general framework for local likelihood density estimation using any p-variate parametric family as the local family, it is evident that the method may struggle in multivariate applications much in the same way as the kernel density estimator does. This is a consequence of the curse of dimensionality, the effect of which is sought remedied by an algorithm provided by Otneim and Tjøstheim (2017). The idea is to fit the p-variate normal distribution $\psi(\mu, \Sigma)$ locally, where μ is the vector of p expectations, and Σ is the $p \times p$ covariance matrix (to which the correlation matrix R corresponds), but under the following structural simplifications:

$$\mu_i(x) = \mu_i(x_1, \ldots, x_p) \overset{\text{def}}{=} \mu_i(x_i) \tag{4}$$

$$\sigma_i(x) = \sigma_i(x_1, \ldots, x_p) \overset{\text{def}}{=} \sigma_i(x_i) \tag{5}$$

$$\rho_{ij}(x) = \rho_{ij}(x_1, \ldots, x_p) \overset{\text{def}}{=} \rho_{ij}\left(x_i, x_j\right). \tag{6}$$

Otneim and Tjøstheim (2017) estimate the local parameters above by first obtaining univariate marginal locally Gaussian fits (eqs. 4 and 5), and then pairwise bivariate locally Gaussian fits (eq. 6). In the second step, the estimates $\widehat{\mu}_i(x_i)$, $\widehat{\mu}_j\left(x_j\right)$, $\widehat{\sigma}_i(x_i)$, and $\widehat{\sigma}_j\left(x_j\right)$ are kept fixed in the estimation of the pairwise local correlation. They argue further that the following transformation technique will produce better density estimates in many situations. The motivation for introducing the simplifications defined in equations 4-6 can be compared to the practical advantages of estimating additive regression models, where $\mathrm{E}(Y) = f(x_1, \ldots, x_p) \overset{\text{def}}{=} f_1(x_1) + \cdots + f_p(x_p)$.

Denote by $F_i(x_i)$, $i = 1, \ldots, p$ the marginal distribution functions of the stochastic vector X, and by $\widehat{F}_i(x_i) = n^{-1}\sum_{i=1}^{n} 1(X_i \leq x_i)$ their empirical counterparts. They then estimate the density $f_Z(z)$ of the vector $Z = \left\{\Phi^{-1}(F_i(X_i))\right\}_{i=1,\ldots,p}$. In practice it is approximated by

$$\widehat{Z} = \left\{\Phi^{-1}\left(\widehat{F}_i(X_i)\right)\right\}_{i=1,\ldots,p}, \tag{7}$$

and where $\Psi(\cdot)$ is the univariate standard normal cdf. In that case, they simplify the estimation problem even further and fix

$$\mu_i(z_i) \overset{\text{def}}{=} 0 \text{ and } \sigma_i(z_i) \overset{\text{def}}{=} 1, \; i = 1, \ldots, p, \tag{8}$$

so that the only parameter functions left to estimate are the pairwise local Gaussian correlations $R(z) = \left\{\rho_{ij}\left(z_i, z_j\right)\right\}_{i<j}$. We use the notation Z, z_i, and z_j to signify that the estimation is performed on the (approximate) standard normal scale or z-scale for short. We can then estimate the joint density $f_Z(z)$ of Z as

$$\widehat{f}_Z(z) = \psi\left(z; \mu(z) = 0, \sigma(z) = 1, R = \widehat{R}(z)\right), \tag{9}$$

where $\mu(z) = \{\mu_i(z)\}$ and $\sigma(z) = \{\sigma_i(z)\}$ for $i = 1, \ldots, p$, and then substitute f_Z for $\widehat{f}_Z$ in the following relation obtained by Otneim and Tjøstheim (2017) in order to estimate the unknown density f_X:

$$f(x) = f_Z(\Phi^{-1}(F_1(x_1)), \ldots, \Phi^{-1}(F_p(x_p))) \times \prod_{i=1}^{p} \frac{f_i(x_i)}{\phi(\Phi^{-1}(F_i(x_i)))}, \tag{10}$$

where $\phi(\cdot)$ is the standard normal pdf. This estimator is implemented the lg package as demonstrated in Section 2.4.1.

One particular feature enjoyed by the jointly normally distributed vector X is that for any partitioning $X = \left(X^{(1)}, X^{(2)}\right)$, the conditional distribution of $X^{(1)}|X^{(2)} = x^{(2)}$ is also normal. In fact, if $X \sim \mathcal{N}(\mu, \Sigma)$, and μ and Σ is partitioned according to $\left(X^{(1)}, X^{(2)}\right)$ as

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix},$$

then $X^{(1)}|X^{(2)} = x^{(2)} \sim \mathcal{N}(\mu^*, \Sigma^*)$, where

$$\mu^* = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}\left(x^{(2)} - \mu_2\right) \tag{11}$$

$$\Sigma^* = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}, \tag{12}$$

see e.g. Johnson and Wichern (2007, chapter 4). Otneim and Tjøstheim (2018) demonstrate that this property may be translated into a corresponding local argument without modification. That is, if the joint density $f_X(\cdot)$ can be written on a locally Gaussian form

$$f_X(x) = \psi(x, \mu(x), \Sigma(x)),$$

then the conditional density of $X^{(1)}|X^{(2)} = x^{(2)}$ can also be written on the same locally Gaussian form with local parameters given by equations (11) and (12), except that all quantities are x-dependent. If we use the transformation technique described above together with simplification (8), the local versions of equations (11) and (12) simplify to

$$\mu^*(z) = R_{12}(z) R_{22}(z)^{-1} z^{(2)}, \tag{13}$$

$$\Sigma^*(z) = R_{11}(z) - R_{12}(z) R_{22}(z)^{-1} R_{21}(z), \tag{14}$$

where we, again, switch to z-notation in order to make it clear that these quantities are estimated on the standard normal z-scale. An estimator $\widehat{f}_{X^{(1)}|X^{(2)}}(\cdot|\cdot)$ of the conditional density $f_{X^{(1)}|X^{(2)}}(\cdot|\cdot)$ follows immediately from an expression corresponding to (10), and the **lg** package provides functions for implementing this estimator in R. We describe the implementation of this functionality in Section 2.4.1.

Finally, we refer to Otneim and Tjøstheim (2021) who take the local version of the conditional covariance matrix (12) (or (14) in the transformed case) as a measure for conditional dependence, and thus as an instrument to test for conditional independence. Consider the stochastic vector $X = \left(X^{(1)}, X^{(2)}, X^{(3)}\right)$, where $X^{(1)}$ and $X^{(2)}$ are scalars and $X^{(3)}$ may be a vector. $X^{(1)}$ is conditionally independent from $X^{(2)}$ given $X^{(3)}$, written $X^{(1)} \perp X^{(2)} \mid X^{(3)}$, if the stochastic variables $X^{(1)} \mid X^{(3)}$ and $X^{(2)} \mid X^{(3)}$ are independent, or, equivalently, if the joint conditional density of $X^{(1)}$ and $X^{(2)}$ given $X^{(3)}$ can be written as the product

$$f_{X^{(1)},X^{(2)}|X^{(3)}}\left(x^{(1)}, x^{(2)}|x^{(3)}\right) = f_{X^{(1)}|X^{(3)}}\left(x^{(1)}|x^{(3)}\right) \times f_{X^{(2)}|X^{(3)}}\left(x^{(2)}|x^{(3)}\right). \tag{15}$$

In this case, denote by $\alpha(z)$ the off-diagonal element in the 2×2 local correlation matrix $R^*(z)$ (which derives directly from $\Sigma^*(z)$ as given in (14)). If X has a local Gaussian distribution, the conditional independence (15) is equivalent to $\alpha(z) \equiv 0$, and Otneim and Tjøstheim (2021) take departures from this relation as evidence against the hypothesis of conditional independence between $X^{(1)}$ and $X^{(2)}$ given $X^{(3)}$. The natural way to quantify this is the test functional

$$T_{n,b}^{CI} = \int h(\widehat{\alpha}(z))\, \mathrm{d}F_n(z). \tag{16}$$

Otneim and Tjøstheim (2021) describe a bootstrap procedure for generating replicates of $T_{n,b}^{CI}$ under the null hypothesis. In Section 2.4.4, we demonstrate how the **lg** package may be used to extract estimates of the local partial correlation and perform tests for conditional independence according to this scheme.

The first step: Creating the lg-object

The local Gaussian correlation may be used to perform a number of statistical analyses, as is evident from the preceding section. The practitioner must first, however, make three quite specific modeling choices; namely (i) to choose an estimation method, i.e., the level of simplification in multivariate applications, (ii) to determine whether the data should be transformed towards marginal standard normality before estimating the LGC, and (iii) to choose a set of bandwidths or at least a method for calculating bandwidths. The architecture of the **lg** package requires the user to make these choices *before* endeavoring further into specific applications by imposing a strict, two-step procedure:

1. Create an lg-object.
2. Apply relevant analysis functions to the lg-object.

In the following, we assume that one has a *data set* `x` loaded into the R workspace, which must be an $n \times p$ matrix (one column per variable, one row per observation), possibly including `NA`s which will be excluded from the analysis, or a data frame having the same dimensions. The fundamental syntax for creating an lg-object is `lg_object <-lg_main(x)`, and we will, in this section, explain how the modeling decisions (i)-(iii) can be encoded into the lg-object by using appropriate arguments in this function.

Selecting the estimation method

Given a data set x having n rows and $p \geq 2$ columns, the user must choose between four distinct estimation methods and specify this choice by using the argument `est_method` to the `lg_main()`-function. We look at the built-in bivariate data set `faithful`, which records the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in the Yellowstone National Park, USA (see the help file in R for more details: `?faithful`), and load the lg package in order to demonstrate the implementation:

```
R> x <- faithful
R> library(lg)
```

1. A full locally Gaussian fit for bivariate data. If $p = 2$, we may fit the bivariate normal $\psi(x, \theta)$ locally to $f(x)$, and by a "full local fit", we mean that we jointly estimate the five parameters

$$\theta(x) = (\mu_1(x_1, x_2), \mu_2(x_1, x_2), \sigma_1(x_1, x_2), \sigma_2(x_1, x_2), \rho(x_1, x_2))$$

by optimizing the local likelihood function (1) in the grid point $x = (x_1, x_2)$. To use this estimation method in the subsequent analysis, specify `est_method = "5par"` in the call to `lg_main()`:

```
R> lg_object <- lg_main(x, est_method = "5par")
```

The resulting `lg_object` is a list of class `lg`, and we may confirm that the assignment has been carried out correctly by inspecting its `est_method`-element:

```
R> lg_object$est_method

[1] "5par"
```

Note that the full locally Gaussian fit for raw data is not available if the number of variables p is greater than 2. The `lg_main()`-function will check for this and print out an error message if $p > 2$ and `est_method = "5par"`.

2. A simplified locally Gaussian fit for multivariate data. As described in the preceding section, we may construct a simplified estimation procedure for calculating the LGC in two steps, which in principle works for any number of dimensions (including $p = 2$):

1. Calculate $\mu_i(x_i)$ and $\sigma_i(x_i)$, $i = 1, \ldots, p$ by fitting the univariate normal distribution locally to each *marginal* density $f_i(x_i)$ of $f(x)$.
2. Keep $\widehat{\mu}_i(x)$ and $\widehat{\sigma}_i(x_i)$, $i = 1, \ldots, p$ from step 1 fixed when estimating $\rho_{ij}\left(x_i, x_j\right)$, $1 \leq i < j \leq p$ by fitting the bivariate normal distribution to each *pair* of variables X_i and X_j.

To use this method, create the lg-object by running the following line:

```
R> lg_object2 <- lg_main(x, est_method = "5par_marginals_fixed")
```

3. A simplified locally Gaussian fit for marginally standard normal data. This estimation method is applicable for marginally standard normal data, or data that have been transformed to approximate marginal standard normality by, e.g., the transformation (7). In that case, we fix the marginal expectation functions and standard deviation functions to the constant values 0 and 1, respectively, and estimate only the pairwise local Gaussian correlations as in (9). To use this estimation method, create the lg-object by running

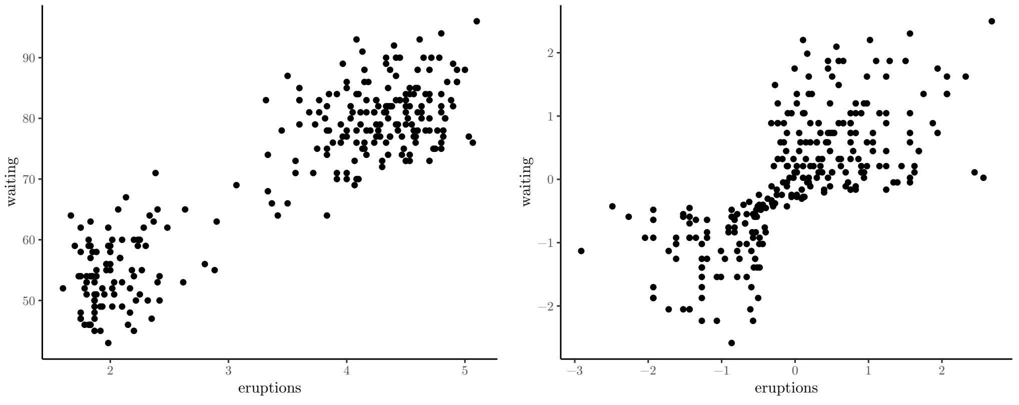
```
R> lg_object3 <- lg_main(x, est_method = "1par")
```

Note that the function call above will issue a warning if the option for transforming the data to marginal standard normality is not at the same time set to `TRUE`, see the next sub-section on data transformation for details.

4. A full locally Gaussian fit for trivariate data. If the number of variables p is equal to 3, and we choose to transform the data to marginal standard normality (see the next sub-section), the transformed density $f_Z(\cdot)$ in (9) may be estimated by jointly estimating the three local correlations $\rho_{12}(z_1, z_2, z_3)$, $\rho_{13}(z_1, z_2, z_3)$, and $\rho_{23}(z_1, z_2, z_3)$. This estimation method was introduced recently by Otneim and Tjøstheim (2021) in order to increase power of their conditional independence test, but it can be used in any application described in this paper that consider trivariate data. To use this estimation method, create the lg-object by running

```
R> lg_object4 <- lg_main(x, est_method = "trivariate")
```

This command will throw an error if the data set x does not have exactly three columns.



(a) The original data.

(b) The same data transformed to marginal standard normality.

Figure 2: The same data on different scales.

Data transformation

Next, the user must determine if the local Gaussian correlation should be estimated directly on the raw data or on the marginally normal pseudo observations (7). This is carried out by using the logical `transform_to_marginal_normality`-argument in `lg_main`, for example:

```
R> lg_object <- lg_main(x, transform_to_marginal_normality = TRUE)
```

The resulting `lg_object` now includes the element `transform_to_marginal_normality` set according to the input, and if this is `TRUE`, it also includes the `transformed_data` and a function `trans_new()` that may be used later to apply the same transformation to, e.g., grid points. If the transformation option is set to `FALSE`, the `transformed_data` element contains the input data `x`, and `trans_new()` is nothing more than the identity mapping for points in $\mathbb{R}^p$. See Figure 2 for two plots that demonstrate the effect of the data transformation on the example data.

Bandwidth selection

Finally, the user must specify a set of bandwidths or a method for calculating them. Given that the different estimation methods described in Section 2.3.1 require different sets of bandwidths (i.e, joint, marginal, and/or pairwise), the easiest approach for the user is to leave the selection and formatting of the bandwidths to the `lg_main()`-function.

The bandwidth plays a slightly different role in local likelihood estimation than elsewhere in the nonparametric literature. It controls the *level of localization* and thus only indirectly the *smoothness* of the estimates. Indeed, suppose we concentrate on the univariate case for the moment and assume that the (single) bandwidth b is small. In that case, we see from the local likelihood function (1) that only the very few observations closest to a fixed grid point x_0 will have significant weight when determining the local parameter estimates $\widehat{\theta}_0(x)$ at that point. Moving on to another nearby point, x_1 may then lead to a fairly different estimate $\widehat{\theta}(x_1)$ because the set of observations having weight in this point is very different. This may, again, lead to rougher parameter estimates $\widehat{\theta}(x)$ and in turn also to rougher density estimates $f\left(x, \widehat{\theta}(x)\right)$.

If the bandwidth b grows large, on the other hand, all observations receive similar weights, and furthermore: the local likelihood function (1) becomes approximately proportional to the ordinary (global) likelihood function $L_n(\theta) = \sum_{i=1}^n \log f(X_i; \theta)$. In other words, the local parameter estimates $\widehat{\theta}(x)$ are smoothed towards the constant maximum likelihood estimates $\widehat{\theta}_{ML}$, and the estimated density $f\left(x; \widehat{\theta}(x)\right)$ towards the maximum likelihood estimate $f\left(x; \widehat{\theta}_{ML}\right)$. This means that the bandwidth may be chosen to reflect the goodness-of-fit of $f(x;\theta)$ to the true density $f(x)$.

In the multivariate applications referred to in this paper, the bandwidth b in (1) is a diagonal matrix, and $1/b$ is naturally taken to represent its inverse.

We have in practice seen two automatic bandwidth selectors employed in the applications referred to in Section 2.2: a cross-validation procedure that is fairly slow to compute but accurate with respect to density estimation, and a plug-in bandwidth that is much quicker to calculate but less accurate with respect to density estimation. We use the argument bw_method to the `lg_main()`-function in order to choose between the two.

1. Choosing bandwidths by cross-validation. The functional

$$CV(b) = -\frac{1}{n}\sum_{i=1}^{n}\log f\left(X_i;\widehat{\theta}^{(-i)}(X_i)\right),$$

where $\widehat{\theta}^{(-i)}(x)$ is the parameter estimate obtained after deleting observation X_i from the data, is proportional to a quantity that estimates the Kullback-Leibler distance between $f\left(\cdot,\widehat{\theta}(\cdot)\right)$ and the true density $f(\cdot)$; see Berentsen and Tjøstheim (2014). The cross-validated bandwidth b_{CV} is hence given by

$$b_{CV} = \arg\min_b CV(b).$$

If we, for example, wish to use the simplified estimation procedure on the transformed data, we need bandwidths for the marginal estimates of the local means and standard deviations, as well as a 2×2 diagonal bandwidth matrix for each pair of variables. This is accomplished by the following call to `lg_main()`:

```
R> # Create the lg-object with bandwidths chosen by cross-validation
R> lg_object <- lg_main(x,
R+                        est_method = "5par_marginals_fixed",
R+                        transform_to_marginal_normality = TRUE,
R+                        bw_method = "cv")
```

The `lg_object` now contains the necessary bandwidths for this configuration, as can be seen by inspecting the contents of its bw-element:

```
R> # Print out the bandwidths
R> lg_object$bw

$marginal
[1] 0.9989327 0.9875333

$marginal_convergence
[1] 0 0

$joint
x1 x2          bw1      bw2 convergence
1  1  2 0.2946889 0.331971           0
```

This is itself a list, containing the crucial elements `marginal` for the p marginal bandwidths, and `joint` that contains the $p(p-1)/2$ bandwidth matrices, one for each pair of variables (which in this bivariate example just *one* variable pair, (`x1, x2`)). The convergence flags stem from the built-in R functions `optim()` and `optimize()` that we use to obtain the minimizer of $CV(\cdot)$, and 0 indicates successful convergence.

2. Using plug-in bandwidths. Obtaining cross-validated bandwidths is unfortunately fairly slow on a standard computer. For sample sizes in the 500-1000 range, the process may take several minutes, which is unfeasible when embarking on analyses that require, e.g., resampling. We have, therefore, implemented a quick plug-in bandwidth selector as well that may suffice in many practical situations, especially at the initial or exploratory stage.

Otneim and Tjøstheim (2017) show that the simplified version of the local Gaussian fit have the same convergence rates as the corresponding nonparametric kernel density estimator for which Silverman (1986) derives the plug-in formula $b = 1.08 \cdot sd(x) \cdot n^{-1/5}$. By specifying `bw_method = "plugin"`, the `lg_main()`-function will select the bandwidths correspondingly, except that the exponent changes to $-1/6$ for joint bandwidths, and the proportionality constant is by default set to 1.75. The latter number is the result of regressing b_{CV} on $n^{-1/6}$ in a large simulation experiment covering various data generating processes (Otneim, 2016). We see the effect of switching to plug-in bandwidths in the code below:

```
R> # Make the lg-object with plugin bandwidths
R> lg_object <- lg_main(x,
R+                        est_method = "5par_marginals_fixed",
R+                        transform_to_marginal_normality = TRUE,
R+                        bw_method = "plugin")
```

Argument	Explanation	Default value
`x`	The data, an $n \times p$ matrix or data frame	
`bw_method`	Method for calculating the bandwidths	`"plugin"`
`est_method`	Estimation method	`"1par"`
`transform_to_ marginal_normality`	Transform the data	`TRUE`
`bw`	The bandwidths to use if already calculated	`NULL`
`plugin_constant_ marginal`	Prop. const. in plugin formula for marg. bw.	1.75
`plugin_exponent_ marginal`	Exponent in plugin formula for marg. bw.	$-1/5$
`plugin_constant_ joint`	Prop. const. in plugin formula for joint bw.	1.75
`plugin_exponent_ joint`	Exponent in plugin formula for joint bw.	$-1/6$
`tol_marginal`	Abs. tolerance when optimizing $CV(b)$, marg.	10^{-3}
`tol_joint`	Abs. tolerance when optimizing $CV(b)$, joint	10^{-3}

Table 1: Arguments to the initialization function `lg_main()`

```
R> # Print out the bandwidths
R> lg_object$bw

$marginal
[1] 0.5703274 0.5703274

$marginal_convergence
[1] NA NA

$joint
  x1 x2       bw1       bw2 convergence
1  1  2 0.6875061 0.6875061          NA
```

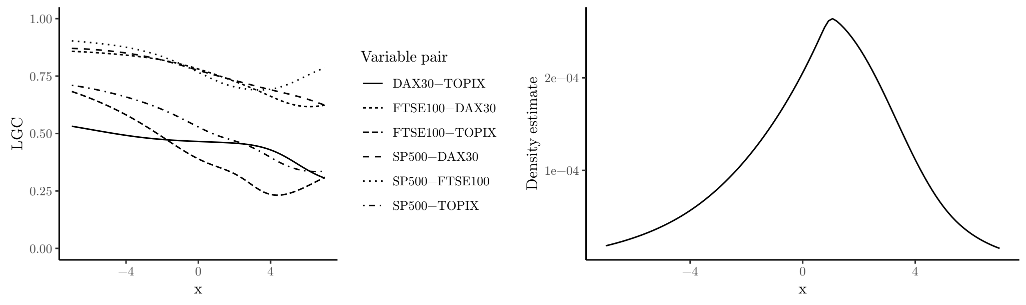
Summary of the initialization function

In the sub-section above, we present the three most important arguments to `lg_main()`. Each of them allows the user to configure one of the three crucial modeling choices. Let us complete this treatment by covering some possibilities to make further adjustments to those choices.

1. The user may supply the bandwidths directly to `lg_main()` by passing them to the bw-argument. They have to be in the correct format, though, which is a list containing the vector `$marginal` if `est_method = "5par_marginals_fixed"`, and always a data frame `$joint` specifying all variable pairs in the `x1` and `x2` columns and the corresponding bandwidths in the `bw1` and `bw2` columns. The function `bw_simple()` will assist in creating bandwidth objects.
2. If `bw_method = "plugin"` the user may change the proportionality constant and exponent in the plugin formula for the joint and, if applicable, the marginal bandwidths. See Table 1 for the necessary argument names.
3. If `bw_method = "cv"`, the user may change the numerical tolerance in the optimization of $CV(b)$. See Table 1 for the necessary argument names.

Statistical inference using the lg package

We proceed in this section to demonstrate how to implement each of the tasks that we discussed in Section 2.2. The general pattern is to pass the lg-object to one of the estimation or test functions provided in the lg package. We will look at some financial data in the examples: the monthly returns on



(a) The estimated pairwise LGC for the stock data along the diagonal indicating that the pairs of stocks are more strongly dependent in the lower part of the distribution than the upper part.

(b) The corresponding joint density estimate of the four return series, plotted along the diagonal in $\mathbb{R}^4$.

Figure 3: Local correlations and density estimates calculated using the `dlg()-function`.

the S&P500, FTSE100, DAX30, and TOPIX stock indices from January 1985 to March 2018 (Datastream, 2018).

Density estimation

We start by introducing a basic function for estimating the LGC on a grid as described by Otneim and Tjøstheim (2017), and thus also a probability density estimate. We create a grid, `x0`, having the same number of columns as the data in the code below. Note that we use the pipe operator `%>%` from the **magrittr** package (Bache and Wickham, 2014) as well as functions from the **dplyr** package (Wickham et al., 2018) for easy manipulation of data frames. We then pass the grid and the lg-object containing our modeling choices to the `dlg()`-function in order to do the estimation.

```
R> # Create an lg-object
R> lg_object <- lg_main(x = stock_data %>% select(-Date),
R+                      est_method = "1par",
R+                      bw_method = "plugin",
R+                      transform_to_marginal_normality = TRUE)
R>
R> # Construct a grid diagonally through the data.
R> grid_size <- 100
R> x0 <- stock_data %>%
R+   select(-Date) %>%
R+   apply(2, function(y) seq(from = -7,
R+                            to = 7,
R+                            length.out = grid_size))
R>
R> # Estimate the local Gaussian correlation on the grid
R> density_object <- dlg(lg_object, grid = x0)
```

The last line of code creates a list containing a number of elements. The two most important are `$loc_cor`, which is a matrix of local correlations having one row per grid point and one column per *pair of variables* (the columns correspond to the rows in `density_object$pairs`), and `$f_est`, which is a vector containing the estimate $\widehat{f}_X(x)$ of the joint density $f_X(x)$ in the grid points specified in `x0`. The estimated local correlations for this example is plotted in Figure 3a, and the corresponding density estimate is plotted (along the diagonal $x_1 = x_2 = x_3 = x_4 = x$) in Figure 3b.

The list `density_object` contains the estimated standard deviations of the local correlations in `$loc_cor_sd`, as well as lower and upper confidence bands `$loc_cor_lower` and `$loc_cor_upper` at the 95% level. We refer to Table 2 for a complete overview of the arguments to `dlg()`.

Note that the configuration `transform_to_marginal_normality = TRUE` and `est_method = 5par` in the bivariate case coincides with the situation considered by Tjøstheim and Hufthammer (2013). In that case, `dlg()` serves as a wrapper for the function `localgauss()` in the **localgauss**-package (Berentsen et al., 2014).

Obtaining the estimate of a conditional density using the Otneim and Tjøstheim (2018) algorithm described in Section 2.2 is very similar. However, one must take particular care of the *ordering* of the variables in the data set. The estimation function, `clg()`, will *always* assume that the free variables

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`grid`	The evaluation points for the LGC	`NULL`
`level`	Level for confidence bands	0.95
`normalization _points`	The estimated density does not integrate to one by construction. `dlg()` will generate the given number of normal variables, having the same moments as the data, approximate $\int \widehat{f}_X(x)\, \mathrm{d}x$ by a Monte Carlo integral, and then normalize the density estimate accordingly	
`NULL`		
`bootstrap`	Calculate bootstrapped confidence intervals instead of asymptotic expressions	`FALSE`
`B`	Number of bootstrap replicates	`500`

Table 2: Arguments to the `dlg()`-function

come first and the conditioning variables last. Let us illustrate this in the following code chunk by estimating the joint conditional density of S&P500 and FTSE100, given that DAX30 = TOPIX = 0.

```
R> # We must make sure that the free variables come first
R> returns1 <- stock_data %>% select(SP500, FTSE100, DAX30, TOPIX)
R>
R> # Create the lg-object
R> lg_object <- lg_main(returns1,
R+                      est_method = "1par",
R+                      bw_method = "plugin",
R+                      transform_to_marginal_normality = TRUE)
R>
R> # Create a grid
R> x0 <- returns1 %>%
R+   select(SP500, FTSE100) %>%
R+   apply(2, function(y) seq(from = -7,
R+                            to = 7,
R+                            length.out = grid_size))
R>
R> # Calculate the conditional density
R> cond_density <- clg(lg_object, grid = x0, cond = c(0, 0))
```

The key argument in the call to `clg()` above is `cond = c(0, 0)`. This means that the last two variables are conditioning variables (and hence, that the first $4 - 2 = 2$ variables are free). The value of the conditioning variables are fixed at DAX30 = 0 and TOPIX = 0, respectively. This also means that the number of columns in the grid `x0` plus the number of elements in `cond` must equal the number of variables p in the data set, and the call to `clg()` will result in an error message if this requirement is not fulfilled. The `clg()`-function takes mostly the same arguments as `dlg()` listed in Table 2, and the conditional density estimate in our example is available in the vector `cond_density$f_est`.

Tests for independence

Three independence tests based on the LGC have appeared in the literature thus far:

1. A test for independence between the stochastic variables X_1 and X_2 based on iid data, cf. Berentsen and Tjøstheim (2014).
2. A test for serial independence between X_t and X_{t-k} within a time series $\{X_t\}$, cf. Lacal and Tjøstheim (2017).
3. A test for serial cross-dependence between X_t and Y_{t-k} for two time series $\{X_t\}$ and $\{Y_t\}$, cf. Lacal and Tjøstheim (2018).

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`h`	The function $h(\cdot)$ in (3)	`function(x) x^2`
`S`	The integration area S in (3). Must be a logical function on potential grid points in $\mathbb{R}^2$	`function(x) as.logical(rep(1, nrow(x)))`
`bootstrap _type`	The bootstrap method, must be either `"plain"`, `"block"` or `"stationary"`	`"plain"`
`block _length`	Block length for the block bootstrap, mean block length for the stationary bootstrap. Calculated by `np::b.star()` (Hayfield and Racine, 2008) if not provided	`NULL`
`n_rep`	Number of bootstrap replicates	1000

Table 3: Arguments to the `ind_test()`-function

As we noted in Section 2.2, their practical implementations are very similar, and the **lg** package provides the function `ind_test()` to perform the tests. Let us first consider the i.i.d. case, and generate 500 observations `test_x` from the well known parabola model $X_2 = X_1^2 + \varepsilon$, where both X_1 and ε are independent and standard normal. In this case, X_1 and X_2 are uncorrelated but obviously strongly dependent. Berentsen and Tjøstheim (2014) considers mainly the full bivariate fit to the raw data, which we easily encode into the lg-object as before. The implementation of the test using 100 bootstrap replicates is shown below.

```
R> # Make the lg-object
R> lg_object <- lg_main(test_x,
R+                      est_method = "5par",
R+                      transform_to_marginal_normality = TRUE)
R> # Perform the independence test
R> test_result <- ind_test(lg_object, n_rep = 100)
R> # Print out the p-value of the test
R> test_result$p_value

[1] 0
```

This may take a few minutes to run on a desktop computer due to bootstrapping. The small p-value indicates that we reject the null hypothesis of independence between X_1 and X_2 in the parabola model defined above. We can further specify the function h and the integration area S in the test statistic (3); see Table 3 for details.

The only difference when testing for serial independence within a time series $\{X_t\}$ is to create a two-column data set consisting of X_t and X_{t-k}. For example, if we wish to perform this test for $k = 1$ for one of the variables in the stock-exchange series, create the matrix of observations as below, and proceed exactly as in the i.i.d. case.

```
R> returns2 <- stock_data %>% select(SP500) %>%
R+   mutate(sp500_lagged = lag(SP500))
```

Finally, the only thing that we must alter in order to perform the third test for serial cross-dependence is the bootstrap method. In the applications above, it suffices to use the standard bootstrap, where we resample with replacement from the data. This is implemented in the `ind_test()`-function by setting the `bootstrap_type`-argument to `"plain"`, which is the default option. When testing for serial cross-dependence, we need to use a block-bootstrap procedure, and Lacal and Tjøstheim (2018) consider two options here: The block bootstrap with either fixed (Kunsch, 1989) or random (Politis and Romano, 1994) block sizes. This choice is specified by choosing `bootstrap_type = "block"` or `bootstrap_type = "stationary"`, respectively, in the call to `ind_test()`. Lacal and Tjøstheim (2018) do not report significant differences in test performance using the different bootstrap types.

Argument	Explanation	Default value
`lg_object_nc`	The lg-object covering the non-crisis period	
`lg_object_c`	The lg-object covering the crisis period	
`grid_range`	Range of diagonal for measuring the LGC	(5%, 95%) quantiles
`grid_length`	The number of grid points to use	30
`n_rep`	Number of bootstrap replicates	1000
`weight`	Weight function	`function(y) rep(1,nrow(y))`

Table 4: Arguments to the `cont_test()`-function

Test for financial contagion

Assume that we observe two financial time series $\{X_t\}$ and $\{Y_t\}$ at times $t = 1, \dots, T$, and that some crisis occurs at time $T^* < T$. As described in Section 2.2, Støve et al. (2014) measure the local correlation between $\{X_t\}$ and $\{Y_t\}$ *before* and *after* T^*, and take significant differences between these measurements as evidence against the null hypothesis of no linkage, or contagion, between the time series. In order to implement this test using the **lg** package, one must create *two* lg-objects: one for the observations covering the non-crisis period and one covering the crisis period. We do not enter a discussion here how to empirically identify such time periods; this is a job that must be done by the practitioner before performing the statistical test.

Let us illustrate the implementation of this test by looking at the same financial returns data that we have used in preceding sections. However, this time we will, in the spirit of Støve et al. (2014), concentrate on *GARCH(1,1)-filtrated daily returns on the S&P500 and FTSE100 indices from 2 January 1985 to 29 April 1987* in order to test for financial contagion between the US and UK stock markets following the global stock market crash of 19 October 1987 ("Black Monday"). Assume that these observations are loaded into the R workspace as the $n_1 \times 2$ data frame `x_nc` containing the observations covering the $n_1 = 728$ days preceding the crisis and the $n_2 \times 2$ data frame `x_c` containing the observations covering the $n_2 = 140$ consecutive trading days starting on *Black Monday* (see the online code supplement for details concerning the GARCH-filtration and data processing). In the code below, we construct one lg-object for each of these data frames with configuration matching the setup used by Støve et al. (2014) and perform the test by means of the `cont_test()`-function.

This function returns a list containing the estimated p-value as well as other useful statistics, including the empirical local correlations measured in the two time periods. See Table 4 for details concerning other arguments that may be passed to this function.

```
R> # Create the two lg-objects
R> lg_object_nc <- lg_main(x_nc,
R+                         est_method = "5par",
R+                         transform_to_marginal_normality = FALSE)
R>
R> lg_object_c <- lg_main(x_c,
R+                         est_method = "5par",
R+                         transform_to_marginal_normality = FALSE)
R>
R> # Run the test with a limited number of bootstrap replicates for
R> # demonstration purposes.
R> result <- cont_test(lg_object_nc, lg_object_c, n_rep = 100)
R>
R> # Print out the p-value
R> result$p_value
```

```
[1] 0.01
```

The small p-value means that we reject the null-hypothesis of no financial contagion between the time series after the crisis.

Partial local correlation

Consider the work finally by Otneim and Tjøstheim (2021), who take the off-diagonal element in the local correlation matrix corresponding to the local conditional covariance matrix (12) or (14) as a local measure of conditional dependence between two stochastic variables $X^{(1)}$ and $X^{(2)}$ given the stochastic vector $X^{(3)}$. Furthermore, in the case with data having been transformed to marginal standard normality, they take the statistic

$$T_{n,b}^{CI} = \int h\left(\widehat{\alpha}\left(z\right)\right) \, \mathrm{d}F_n\left(z\right) \tag{17}$$

as a measure of *global* conditional dependence. The **lg** package provides two key functions in this framework. The first, `partial_cor()`, calculates the local partial correlations as well as their estimated standard deviations on a specified grid in the $\left(x^{(1)}, x^{(2)}, x^{(3)}\right)$-space, and is essentially a wrapper function for the `clg()`-function presented in Section 2.4.1. See Table 5 for details. The second function, `ci_test()`, performs a test for conditional independence between the first two variables in a data set given the remaining variables using the test statistic (17) and a special bootstrap procedure (described briefly below) for approximating the null distribution.

It is well known in the econometrics literature that conditional independence tests are instrumental in the empirical detection of Granger causality (Granger, 1980). For example, if we continue to concentrate on the monthly stock returns data that we have already loaded into memory, we may test whether

$$\mathrm{H}_0: \qquad R_{\mathrm{FTSE100},t} \perp R_{\mathrm{SP500},t-1} \mid R_{\mathrm{FTSE100},t-1} \tag{18}$$

in the period starting in January 2009, the converse of which is a sufficient, but not necessary, condition for $R_{\mathrm{SP500},t}$ Granger causing $R_{\mathrm{FTSE100},t}$. We perform the test by running the code below, where `x` is a data frame having the following columns strictly ordered as $R_{\mathrm{FTSE100},t}$, $R_{\mathrm{SP500},t-1}$, and $R_{\mathrm{FTSE100},t-1}$ (see the online supplement for the pre-processing of data).

The critical values of this test are calculated using the bootstrap under the null hypothesis by independently resampling replicates from the conditional density estimates $\widehat{f}_{X^{(1)}|X^{(3)}}\left(x^{(1)}|x^{(3)}\right)$ and $\widehat{f}_{X^{(2)}|X^{(3)}}\left(x^{(2)}|x^{(3)}\right)$, as obtained by the `clg()`-function, using an approximated accept-reject algorithm. In order to avoid excessive optimization of the local likelihood function (1), we estimate $f_{X^{(1)}|X^{(3)}}$ and $f_{X^{(2)}|X^{(3)}}$ on the univariate regular grids $x_0^{(1)}$ and $x_0^{(2)}$, respectively (while keeping $x^{(3)}$ fixed at the observed values of $X^{(3)}$), and produce interpolating functions $\tilde{f}_{X^{(1)}|X^{(3)}}$ and $\tilde{f}_{X^{(2)}|X^{(3)}}$ using cubic splines. It is much less computationally intensive to generate replicates from $\tilde{f}$ than directly from $\widehat{f}$.

We refer to the documentation of the **lg** package for details on how to finely tune the behavior of the bootstrapping algorithm by altering the arguments of the `ci_test()`-function and limiting our treatment to describing the arguments most suitable for modifications by the user in Table 6.

```
R> # Create the lg-object
R> lg_object <- lg_main(returns4)
R>
R> # Perform the test
R> test_result <- ci_test(lg_object, n_rep = 100)
R>
R> # Print out result
R> test_result$p_value

[1] 0.51
```

The conditional independence test does not provide evidence against the null-hypothesis (18).

Graphics

We conclude this article by describing the `corplot()` function for drawing dependence maps such as those displayed in Figure 1. Berentsen et al. (2014) report on such capabilities in the **localgauss** package, but the possibility of creating dependence maps was unfortunately removed from **localgauss** in the latest version 0.4.0 due to incompatibilities with the **ggplot2** (Wickham, 2016) plotting engine. We make up for this loss by providing `corplot()`, a function that plots the estimated local correlations as provided by `dlg()`, or the estimated local *partial* correlations as provided by `partial_cor()`.

The plotting function is highly customizable and provides a number of options covering most basic graphical options. Users well versed in the **ggplot2** package may also modify the graphical object returned by `corplot()` in the standard way by adding layers as demonstrated in the example below.

In the first example, we generate a set of bivariate normally distributed data using the **mvtnorm** package (Genz et al., 2018) and estimate the local Gaussian correlation on a regular `grid` using the `dlg()`-function. Passing the resulting `dlg_object` to `corplot()` without further arguments results in Figure 4.

```
R> # Make a regular grid in the domain of the distribution
R> grid <- expand.grid(seq(-3, 3, length.out = 7),
R+                     seq(-3, 3, length.out = 7))
R>
R> x <- mvtnorm::rmvnorm(500, sigma = matrix(c(1, rho, rho, 1), 2))
R> lg_object <- lg_main(x,
R+                      est_method = "5par",
R+                      transform_to_marginal_normality = FALSE,
R+                      plugin_constant_joint = 4)
R> dlg_object <- dlg(lg_object, grid = grid)
R>
R> # Make a dependence map using default setup
R> corplot(dlg_object)
```

We may tweak the appearance of our dependence map by passing further arguments to `corplot()`. Some of the options are demonstrated in the code chunk below, in which we, for example, superimpose the observations (by setting `plot_obs = TRUE`) and preventing the estimated local correlations from being plotted in areas without data. The latter option is available through the argument `plot_thres`, which works by calculating a bivariate kernel density estimate $\tilde{f}(x_1, x_2)$ for the pair of variables in question and only allowing $\widehat{\rho}(x_1, x_2)$ to be plotted if $\tilde{f}(x_1, x_2) / \max \tilde{f}(\cdot) >$ `plot_thres`. Adding layers to a dependence map using the ordinary **ggplot2** syntax works as well, which we demonstrate in Figure 5 by changing the **ggplot2** theme.

The plotting function works in the same way when plotting the local partial correlations returned by `partial_cor()`, and the arguments of `corplot()` are summarized in Table 7.

```
R> corplot(dlg_object1,
R+         plot_obs = TRUE,
R+         plot_thres = 0.01,
R+         plot_labels = FALSE,
R+         alpha_point = 0.3,
R+         main = "",
R+         xlab = "",
R+         ylab = "") +
R+   theme_classic()
```

Conclusion

The statistical literature has seen a number of applications of local Gaussian approximations in the last decade, covering several topics in dependence modeling and inference, as well as the estimation of multivariate density and conditional density functions. In this paper, we demonstrate the implementation of these methods in the R programming language using the **lg** package, as well as the graphical representation of the estimated local Gaussian correlation. The package is complete in the sense that all major methods that have been published within this framework is now easily accessible to the practitioner. The package is also designed with a modular infrastructure that allows future methodological developments using local Gaussian approximations to be easily added to the package.

Acknowledgements

The author gratefully acknowledges the constructive comments made by two anonymous referees.

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`grid`	The evaluation points for the LGPC, **must** be a data frame or matrix having 2 columns	`NULL`
`cond`	Vector with fixed values for $X^{(3)}$	`NULL`
`level`	Significance level for approximated confidence bands	0.95

Table 5: Arguments to the `partial_cor()-function`

Argument	Explanation	Default value
`lg_object`	The lg-object created by `lg_main()`	
`h`	The function $h(\cdot)$ in (16)	`function(x) x^2`
`n_rep`	Number of bootstrap replicates	500

Table 6: Arguments to the `ci_test()-function`

Argument	Explanation	Default value
`dlg_object`	The object created by `dlg()` or `partial_cor()`	
`pair`	Which pair to plot if more than two variables	1
`gaussian_scale`	Logical. Plot on the marginal st. normal scale?	`FALSE`
`plot_colormap`	Logical. Plot the colormap?	`TRUE`
`plot_obs`	Logical. Superimpose observations?	`FALSE`
`plot_labels`	Logical. Plot labels on dependence map?	`TRUE`
`plot_legend`	Logical. Add legend?	`FALSE`
`plot_thres`	Threshold for plotting the estimated LGC	0
`alpha_tile`	Transparency of color tiles	0.8
`alpha_point`	Transparency of points	0.8
`low_color`	Color representing $\widehat{\rho} = -1$	`"blue"`
`high_color`	Color representing $\widehat{\rho} = +1$	`"red"`
`break_int`	Break interval for color coding	0.2
`label_size`	Size of labels in plot	3
`font_family`	Font family for labels	`"sans"`
`point_size`	Size of points, if plotted	`NULL`
`xlim`, `ylim`	Axis limits	`NULL`
`xlab`, `ylab`	Axis labels	`NULL`
`rholab`	Title of legend	`NULL`
`main`, `subtitle`	Title and subtitle of plot	`NULL`

Table 7: Arguments to the `corplot()`-function

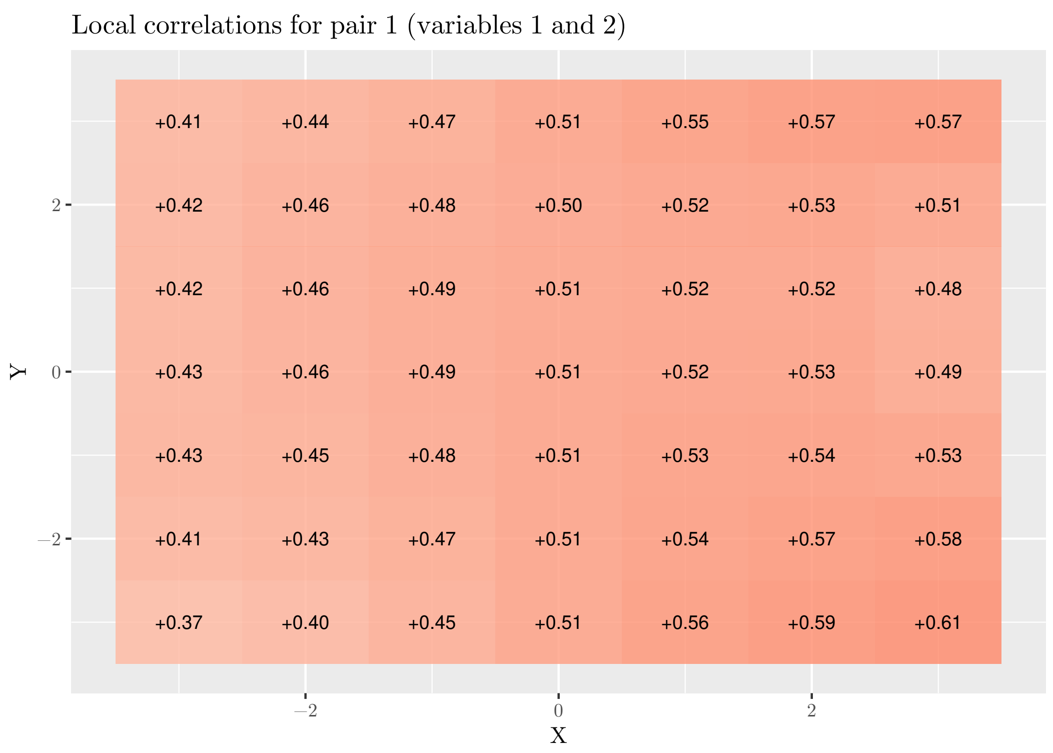


Figure 4: Dependence map produced by corplot() using the default configuration

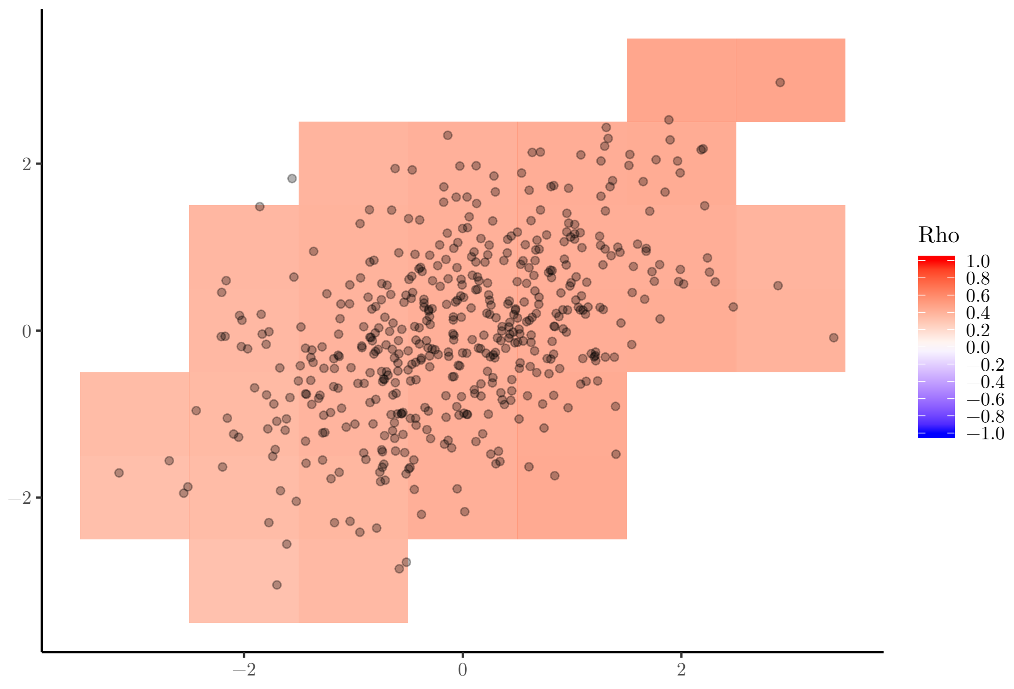


Figure 5: Dependence map produced by tweaking the arguments of corplot()

Bibliography

S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. URL https://CRAN.R-project.org/package=magrittr. R package version 1.5. [p47]

G. D. Berentsen and D. Tjøstheim. Recognizing and visualizing departures from independence in bivariate data using local Gaussian correlation. *Statistics and Computing*, 24(5):785–801, 2014. URL https://doi.org/10.1007/s11222-013-9402-8. [p38, 40, 45, 48, 49]

G. D. Berentsen, T. Kleppe, and D. Tjøstheim. Introducing localgauss, an R-package for estimating and visualizing local Gaussian correlation. *Journal of Statistical Software*, 56(12):1–18, 2014. URL https://doi.org/10.18637/jss.v056.i12. [p38, 47, 51]

Datastream, 2018. [p40, 47]

K. J. Forbes and R. Rigobon. No contagion, only interdependence: measuring stock market co-movements. *The Journal of Finance*, 57(5):2223–2261, 2002. URL https://doi.org/10.1111/0022-1082.00494. [p40]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2018. URL https://CRAN.R-project.org/package=mvtnorm. R package version 1.0-8. [p52]

C. W. Granger. Testing for causality: a personal viewpoint. *Journal of Economic Dynamics and control*, 2: 329–352, 1980. URL https://doi.org/10.1016/0165-1889(80)90069-X. [p51]

T. Hayfield and J. S. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5), 2008. URL https://doi.org/10.3929/ethz-b-000073064. [p49]

N. Hjort and M. Jones. Locally parametric nonparametric density estimation. *Annals of Statistics*, 24(4): 1619–1647, 1996. URL https://doi.org/10.1214/aos/1032298288. [p39, 40]

R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis, Sixth Edition*. Pearson Education Iternational, 2007. [p41]

L. A. Jordanger. *localgaussSpec: An R-package for local Gaussian spectral analysis*, 2018. URL https://github.com/LAJordanger/localgaussSpec. R package. [p38]

L. A. Jordanger and D. Tjøstheim. Nonlinear spectral analysis: A local gaussian approach. *Journal of the American Statistical Association*, pages 1–55, 2020. [p38]

H. R. Kunsch. The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, 17(3):1217–1241, 1989. URL https://doi.org/10.1214/aos/1176347265. [p49]

V. Lacal and D. Tjøstheim. Local Gaussian autocorrelation and tests of serial dependence. *Journal of Time Series Analysis*, 38(1):51–71, 2017. [p38, 40, 48]

V. Lacal and D. Tjøstheim. Estimating and testing nonlinear local dependence between two time series. *Journal of Business & Economic Statistics*, pages 1–13, 2018. [p38, 40, 48, 49]

H. Otneim. *Multivariate and conditional density estimation using local Gaussian approximations*. PhD thesis, University of Bergen, 2016. [p45]

H. Otneim. *lg: Locally Gaussian Distributions: Estimation and Methods*, 2019. URL https://CRAN.R-project.org/package=lg. R package version 0.4.0. [p38]

H. Otneim and D. Tjøstheim. The locally Gaussian density estimator for multivariate data. *Statistics and Computing*, 27(6):1595–1616, 2017. [p38, 40, 41, 45, 47]

H. Otneim and D. Tjøstheim. Conditional density estimation using the local Gaussian correlation. *Statistics and Computing*, 28(2):303–321, 2018. [p38, 41, 47]

H. Otneim and D. Tjøstheim. The locally Gaussian partial correlation. *Journal of Business & Economic Statistics*, pages 1–33, 2021. [p38, 42, 43, 51]

D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994. [p49]

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986. [p38, 45]

B. Støve, D. Tjøstheim, and K. Hufthammer. Using local Gaussian correlation in a nonlinear re-examination of financial contagion. *Journal of Empirical Finance*, 25:785–801, 2014. [p38, 40, 50]

D. Tjøstheim and K. O. Hufthammer. Local Gaussian correlation: A new measure of dependence. *Journal of Econometrics*, 172(1):33–48, 2013. [p38, 39, 47]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL http://ggplot2.org. [p51]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.6. [p47]

Håkon Otneim
Department of Business and Management Science,
NHH Norwegian School of Economics,
Helleveien 30, 5045 BERGEN
Norway
ORCID: 0000-0002-6004-0237
hakon.otneim@nhh.no

Estimating Social Influence Effects in Networks Using A Latent Space Adjusted Approach in R

by Ran Xu

Abstract Social influence effects have been extensively studied in various empirical network research. However, many challenges remain in estimating social influence effects in networks, as influence effects are often entangled with other factors, such as homophily in the selection process and the common social-environmental factors that individuals are embedded in. Methods currently available either do not solve these problems or require stringent assumptions. Recent works by Xu (2018) and others have shown that a latent space adjusted approach based on the latent space model has the potential to disentangle the influence effects from other processes, and the simulation evidence has shown that this approach outperforms other state-of-the-art approaches in terms of recovering the true social influence effect when there is an unobserved trait co-determining influence and selection. In this paper, I will further illustrate how the latent space adjusted approach can account for bias in the estimation of social influence effects and how this approach can be easily implemented in R.

Introduction

Social influence effects, sometimes referred to as spillover or contagion effects, have long been central to the field of social science (Asch, 1972; Erbring and Young, 1979; Bandura, 1986). It is defined as the propensity for the behavior of an individual to vary along with the prevalence of that behavior in some reference group (Manski, 1993), such as one's social contacts. With the availability of social network data, social influence effects have received much attention and have been widely used to study various phenomena such as the spread of health behavior (e.g., obesity and smoking) (Christakis and Fowler, 2007, 2008), psychological states (Cacioppo et al., 2008; German et al., 2012), professional practices (Frank et al., 2004) and information diffusion (Valente, 1995, 1996).

However, many challenges remain in estimating social influence effects, especially from observational network data, because it is difficult to separate the effect of social influences from other processes that operate simultaneously. That is, when we observe that people in close relationships or interactions tend to be similar in their behaviors or states, it is difficult to identify the underlying mechanisms that generate these patterns. One mechanism could be influence or contagion (Friedkin and Johnsen, 1999; Friedkin, 2001; Oetting and Donnermeyer, 1998), whereby individuals assimilate the behavior of their network partners. Another mechanism could be selection – in particular, homophily (Mcpherson and Smith-Lovin, 1987; Mcpherson et al., 2001), in which individuals seek to interact with similar others. Furthermore, there could be some common social-environmental factors – individuals with previous similarities select themselves into the same social settings (e.g., hospital or alcoholics anonymous (AA) support group), and actual network formation just reflects the opportunities of meeting in this social setting (Feld, 1981, 1982; Kalmijn and Flap, 2001)[1].

Entanglement among these different mechanisms unavoidably induces bias when we estimate social influence effects (Shalizi and Thomas, 2011). Various statistical methods and recent advancements in the field of social network analysis have attempted to reduce the bias in estimating social influence effects, such as instrumental variable (IV) methods (Bramoulle et al., 2009), propensity score methods (Aral et al., 2009), and stochastic actor-oriented models (SAOM) (Snijders et al., 2010). Although each potentially leverages extra information in the data to reduce bias, none can claim to eliminate all sources of bias.

Recent works by Xu (2018) and others (Shalizi and McFowland III, 2018) have shown that a latent space adjusted approach based on the latent space model (Hoff et al., 2002) has the potential to disentangle the social influence effects from other processes operating at the same time, and simulation evidence has shown that this approach outperforms some other state-of-the-art methods (e.g., instrumental variable method, structural equation model) in terms of recovering the true social influence effects. In this paper, I will illustrate how the latent space adjusted approach can account for

[1]There are also structural constraints such as transitivity and preferential attachment which could cause people to become friends. However, these mechanisms in themselves do not entangle with influence (e.g., one can befriend another having high popularity but different behavior). In these cases, another mechanism must be present to induce similarities between these friends (e.g., selection of common friends based on similarity in attributes), and thus the entanglement goes back to the original three mechanisms, namely influence, selection based on homophily, and social-environmental factors.

bias in the estimation of social influence effects and demonstrate how it can be easily incorporated with various models in R to estimate social influence effects. In the following sections, I will first explain the challenges in estimating social influence effects and how they can be framed as an omitted variable bias problem. Then I will formally introduce the latent space adjusted approach and explain how it can account for bias in the estimation of social influence effects. Finally, I will demonstrate how this approach can be easily implemented in R and how it can be incorporated with various models to estimate social influence effects, including a dynamic linear-in-mean influence model and a stochastic actor-oriented model (SAOM).

Identification of Social Influence as An Omitted Variable Bias Problem

Similarities of behavior, state, and characteristics of two individuals in a network relationship can be caused by three primary mechanisms: influence, homophilous selection, or common social-environmental factors (Vanderweele and An, 2013). While it is possible to rule out some mechanisms through random treatment assignment or networks in experiments, entanglement among these different mechanisms makes it difficult to correctly estimate social influence effects from observational data (Xu, 2020). The challenges in estimation caused by entanglement among social influence effects and common social-environmental factors can be easily framed as an omitted variable bias problem (e.g., ignoring the group or environment individuals belong to when estimating the social influence model). What is less obvious is that entanglement between the influence and the homophilous selection can also be framed as an omitted variable bias problem. As pointed out by Steglich et al. (2010), one of the important concerns of SAOM is the "possibility that there may be non-observed variables co-determining the probabilities of change in network and/or behavior". Shalizi and Thomas (2011) have shown that when there is an unobserved trait that co-determines both behavior and network choice, social influence effects are generally unidentifiable as social influence and homophily (selection) are generically confounded through this unobserved trait.

To give an example, assuming that adolescent i's alcohol use at time t, $alcohol_{it}$, is the outcome of interest, and it is a function of his/her previous alcohol use, $alcohol_{it-1}$, his/her friend j's previous alcohol use, $alcohol_{jt-1}$ (i.e., social influence), and an unobserved tendency for substance-abuse (arrow D in Figure 1). At the same time, there is a homophilous selection based on this unobserved substance-abuse tendency in the network – individuals with similar levels of substance-abuse tendency are more likely to be friends (arrow A in Figure 1). As a result, person j's alcohol use, which is a function of person j's substance-abuse tendency (arrow B_j), will be correlated with person i's substance-abuse tendency through homophilous selection (arrow C in Figure 1). However, as the substance-abuse tendency is unobserved, this violates the key assumption of most estimation methods (i.e., the omitted variable should not correlate with the independent variables) such that the estimates of the social influence effects will be biased and inconsistent.

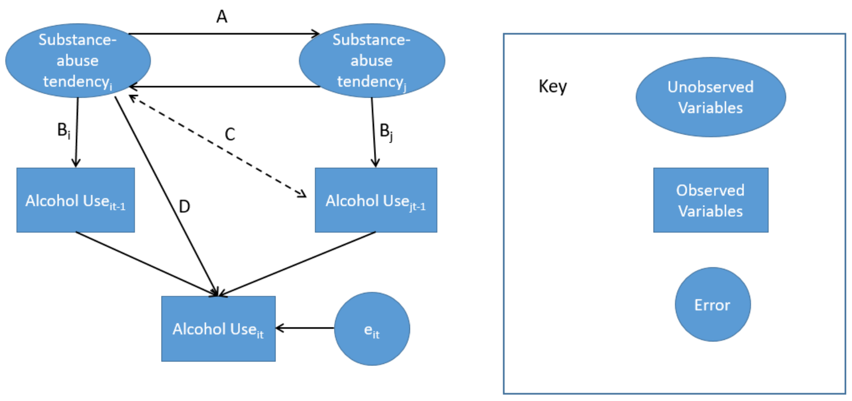


Figure 1: Demonstration of the omitted variable bias.

Latent Space Adjusted Approach

Xu (2018) recently proposed a latent space adjusted approach, and simulation evidence has shown that it has the potential to correctly identify social influence effects when there is an unobserved variable that co-determines the influence and the selection process. Specifically, the behavioral (influence) model can be represented as

$$Y_i = f(Z_{ij}, Y_j, X_i, c_i) \tag{1}$$

where the behavior of person i, Y_i, is a function of the behaviors of his/her network partners j, Y_j, the network relations between i and j, Z_{ij}, person i's observed characteristics X_i as well as a time-invariant unobserved trait c_i.[2] For example, adolescent i's alcohol use at time t, Y_{it}, can be a function of his/her previous alcohol use, Y_{it-1}, his/her close friends' previous alcohol use Y_{jt-1}, his/her own cigarette use, X_{it}, and a time-invariant unobserved tendency for substance abuse c_i.

The selection model can be represented as

$$P(Z_{ij} = 1) = g(X_{ij}, D(c_i, c_j)) \tag{2}$$

where the probability that person i and person j has a network relation is a function of the individual and dyadic level observed variables X_{ij}, and a distance function of the unobserved trait c between i and j, such that i and j are more likely to have a network relation when they are close to each other in terms of c. For example, the probability that adolescent i and j are friends at time t Z_{ijt} can be a function of the absolute value of the differences between their cigarette use at time t $|X_{it} - X_{jt}|$ (observed homophily) and the absolute value of the differences between their unobserved tendencies for substance abuse $|c_i - c_j|$ (latent homophily), where i and j are more likely to become friends when they are similar to each other in terms of the cigarette use (X) or the tendency for substance abuse (c).

Ideally, if there is any information about this unobserved trait c from the selection process in (2), it can be leveraged and used in the estimation of the behavioral model in (1), and in principle this will reduce the bias when estimating the social influence effects. However, the estimations of most selection models are based on the observed variables and thus do not attend to those factors that are unobserved. Xu (2018) extended this idea and built on the theoretical logic of latent space models (Hoff et al., 2002). Latent space models assume that each individual has a "latent position" that lies in an unobserved n-dimensional social space, and the probability of interaction between any two actors depends on the latent positions of these two actors. Specifically, the model takes a logistic form and can be specified as

$$logodds(Z_{ij} = 1 \| c_i, c_j, x_{ij}, \alpha, \beta) = \alpha + \beta' x_{ij} - |c_i - c_j| \tag{3}$$

Here, Z_{ij} indicates whether there is a network relation from i to j, x_{ij} is a vector of the observed covariates (at the dyadic level or node level), c indicates the latent social position of i and j, and $|c_i - c_j|$ represents the Euclidean distance between i and j's latent positions (it could also be replaced by other distance functions). When i and j are closer to each other in terms of the latent position c, they will have a higher probability of having a network relation. And these latent positions can represent determinants of the network relations that have not been accounted for by the observed variables in the selection process. The parameters α and β are estimated using either Maximum-Likelihood Estimation (MLE) or Markov Chain Monte Carlo (MCMC) methods, and the latent positions c can be estimated by Minimum Kullback-Leibler (MKL) estimates (Shortreed et al., 2006).

It is not difficult to see that the latent space model in (3) is very similar to the selection process in (2), except that c represents the latent position in the latent space model, while c represents the individual's unobserved trait in (2).[3] For any pair of i and j, a smaller distance between the latent social positions or the unobserved traits will result in a higher likelihood of having network relations. Therefore, when two individuals are close to each other in terms of the unobserved traits, they are more likely to have a network relation, and they should also be close to each other in terms of the latent positions (and vice versa).

Furthermore, if these latent positions from the latent space model are estimated accurately enough, the estimates of these latent positions can be used as the proxies for the unobserved traits that determine the homophily in the selection process. In fact, for two one dimensional variables X and Y, if the distance correlation (e.g., correlation between $|X_i - X_j|$ and $|Y_i - Y_j|$) is 1, then Y can be written as a linear function of X: $Y = a + bX$ (Szekely et al., 2007), which means the correlation between the two variables are either 1 or -1. **Thus, the estimated latent positions from the latent space model can be used as the proxies (Wooldridge, 2011) for the unobserved traits that co-determine influence and selection, and including the latent positions as additional covariates in the behavioral model will reduce the bias in the estimation of social influence effects**. For example, to model adolescents' social influences on their alcohol use, we can first use a latent space model to model the friendship network of adolescents and acquire the estimated "latent positions" for each individual, and then use

[2]Here Y, X, Z are assumed to be time-variant and c is assumed to be time-invariant, but the assumption can be relaxed.

[3]Here I only choose one-dimensional latent social positions to mimic the unobserved trait that drives the homophily in the selection process. The arguments can easily be extended to multi-dimensional latent positions.

these estimates as the proxies for the unobserved substance-abuse tendencies in the behavioral model, and thus achieve a better estimation of the true social influence effects. If the social network data is longitudinal, estimated latent positions from each time point can be included as separate covariates in the behavioral model to better approximate the unobserved trait.

Shalizi and McFowland III (2018) have shown that if the network grows according to a continuous latent space model, the latent positions can be consistently estimated. Controlling for these latent positions allows for unbiased and consistent estimation of the social-influence effects in additive influence models. Simulation evidence from Xu (2018) has shown that when there is a time-invariant unobserved variable that co-determines selection and influence, the estimated latent positions can be good proxies for the unobserved variable, and the latent space adjusted approach outperforms other methods that are commonly used to deal with the unobserved variables, including a structural equation based estimator (implemented using **lavaan** package in R (Rosseel and Jorgensen, 2019)) and an instrumental variable estimator (implemented using **plm** package in R (Croissant et al., 2021)), in producing the smallest bias and standard error of the social influence effect using a dynamic linear-in-mean influence model. The results are robust to the inclusion of additional covariates, structural properties (e.g., transitivity) in networks, different scaling of the latent space model, or even misspecifications (Xu, 2018).

Finally, there are a couple of things to note: (1) for the estimated latent positions to better approximate the unobserved traits, we need to control for other mechanisms that are likely to drive the selection process in the latent space model, such as homophily based on the observed variables, transitivity, alter, and ego effects. (2) In principle this method can apply to any functional form of the behavioral/influence model (e.g., stochastic actor-oriented models), as essentially this approach just adds additional covariates as the proxies for the unobserved traits. (3) As the scales and the actual positions of the estimated latent positions are essentially arbitrary (Hoff et al., 2002), the actual values of the latent positions might be very different from the actual values of the unobserved traits that co-determine influence and selection. However, as long as the estimated latent positions are highly correlated with the unobserved traits (i.e., actors who are close to each other on the latent positions are also close to each other in terms of the unobserved traits), the social influence effects can still be consistently estimated. (4) This approach works in scenarios where there are unobserved traits that co-determine influence and selection (homophily).[4] It does not improve the estimation of social influence effects when the unobserved traits are only present in one process but not the other.

An Empirical Example in R

In this section, I present an empirical example illustrating how to implement the latent space adjusted approach to estimate the social influence effect using R 3.5.2. The data comes from the social network data collected in the Teenage Friends and Lifestyle Study data set (Michell, 2000; Pearson and West, 2003). Friendship network data and substance use were recorded for a cohort of 50 female pupils in a school in the West of Scotland. The panel data were recorded over three years, starting in 1995, when the pupils were aged 13, and ending in 1997. The friendship networks were formed by allowing the pupils to name up to twelve best friends. Pupils were also asked about substance use and adolescent behavior associated with, for instance, lifestyle, sporting behavior, tobacco, alcohol, and cannabis consumption. The question on sporting activity asked if the pupil regularly took part in any sport, or went training for sport, out of school (e.g., football, gymnastics, skating, mountain biking). The school was representative of others in the region in terms of social class composition (Pearson and West, 2003). The key variables used in this example were measured three times from 1995-1997 and included pupils' friendship networks (binary variable representing each possible directed pair, 1 if nominated and 0 otherwise), smoking (measured on a 1-3 scale), drug use (measured on a 1-4 scale), alcohol use (measured on a 1-5 scale) and sport activity (measured on a 1-2 scale). The dataset is available here.

First, I install and load all the packages needed in R. **latentnet** is the package that is used to estimate the latent space model (Krivitsky and Handcock, 2020). And **statnet** is the package to manipulate and create the network object (Handcock et al., 2019).

```
> library(latentnet)
> library(RSiena)
> library(sna)
> library(statnet)
```

The network data comes with the **RSiena** package (Ripley et al., 2018). I load the attribute data into the current session and create network objects over 3 time points:

[4]In principle this approach could also account for unobserved social-environmental factors that drive influence and selection.

```
##Load girls' attributes on smoking, drug use, sport and alcohol use
> s50s<-read.table("s50-smoke.dat",header=FALSE)
> s50d<-read.table("s50-drugs.dat",header=FALSE)
> s50sp<-read.table("s50-sport.dat",header=FALSE)
> s50a<-read.table("s50-alcohol.dat", header=FALSE)

## Create network object with attributes for each time point
> g1<-network(s501,directed=TRUE)
> g1%v%"a" <- s50a[,1]
> g1%v%"s" <- s50s[,1]
> g1%v%"sp" <- s50sp[,1]
> g1%v%"d" <- s50d[,1]

> g2<-network(s502,directed=TRUE)
> g2%v%"a" <- s50a[,2]
> g2%v%"s" <- s50s[,2]
> g2%v%"sp" <- s50sp[,2]
> g2%v%"d" <- s50d[,2]

> g3<-network(s503,directed=TRUE)
> g3%v%"a" <- s50a[,3]
> g3%v%"s" <- s50s[,3]
> g3%v%"sp" <- s50sp[,3]
> g3%v%"d" <- s50d[,3]
```

We can plot each network and observe how they have changed over time. Figure 2 shows how these girls' friendship networks have changed from 1995 to 1997. The network graphs show that there have been considerable network changes over time, and distinct components/clusters have emerged over time.

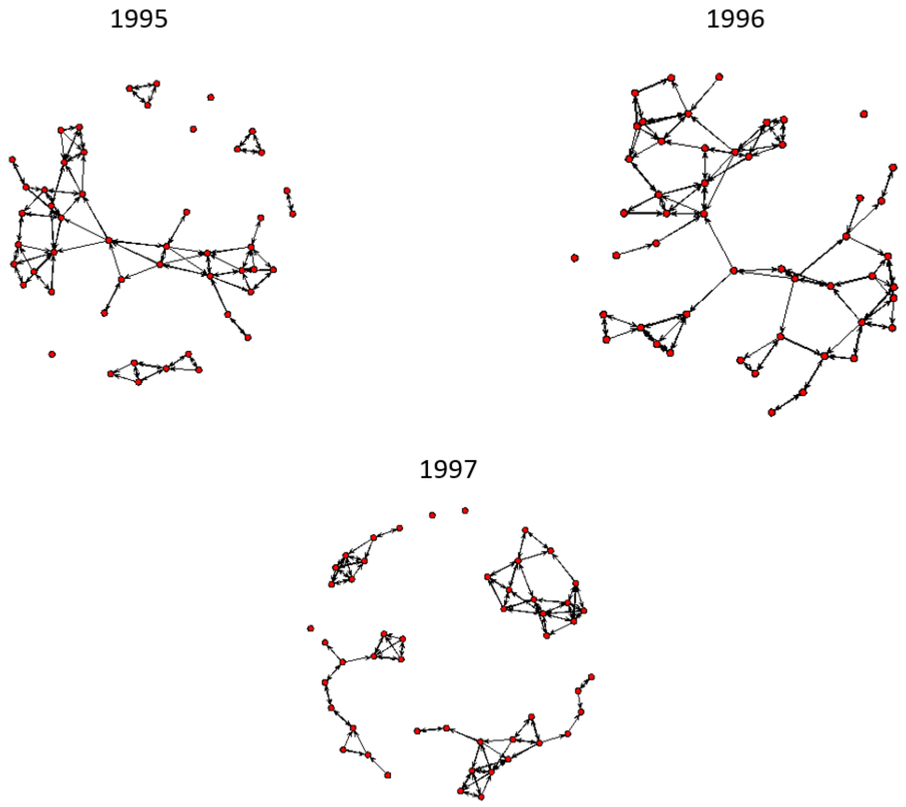


Figure 2: Girls' friendship network from 1995 to 1997.

My primary research question is whether these girls influence each other's alcohol use. Here I demonstrate how to estimate the social influence effect by incorporating the latent space adjusted approach with a dynamic linear-in-mean model (Friedkin and Johnsen, 1990) using the "lm" function and a stochastic actor-oriented model using **RSiena** package (Snijders et al., 2010; Ripley et al., 2018) in R. I start by estimating the latent space models using the "ergmm" function to extract the estimated latent positions. Specifically, I estimate two latent space models based on networks in 1995 and 1996 with one dimensional latent space, while controlling for homophily based on observed variables such as alcohol, smoking, drug use and sport:

```
> m1<-ergmm(g1 ~ euclidean(d = 1)+absdiff("a")+absdiff("s")+absdiff("sp")+absdiff("d"),
+ control=ergmm.control(sample.size=5000,burnin=20000,interval=10,Z.delta=5))
> m2<-ergmm(g2 ~ euclidean(d = 1)+absdiff("a")+absdiff("s")+absdiff("sp")+absdiff("d"),
+ control=ergmm.control(sample.size=5000,burnin=20000,interval=10,Z.delta=5))
```

Once the latent space models are estimated, I can extract the latent positions and add them as additional covariates when estimating the behavioral/influence model. First I estimate a dynamic linear-in-mean influence model, which can be represented as (Friedkin and Johnsen, 1990):

$$Y_{it} = \beta_0 + \beta_1 Y_{it-1} + \beta_2 \frac{\sum Z_{ijt-1} Y_{jt-1}}{\sum Z_{ijt-1}} + \beta_3 X_{it} + e_{it}, \tag{4}$$

where Y_{it} is the behavior of i at time t, Y_{it-1} is the previous behavior of i, Z_{ijt-1} is a dummy variable indicating if there is a link from i to j at time $t-1$, i.e., 1 if yes and 0 otherwise, and $\frac{\sum Z_{ijt-1} Y_{jt-1}}{\sum Z_{ijt-1}}$ is the average behaviors at time $t-1$ among the network neighbors of i, and β_2 represents the social influence effect. X_{it} represents other concurrent variables of i that might affect the behavioral outcome Y. To estimate the dynamic linear-in-mean influence model, I first need to construct the dataset used by this model:

```
## create the average alcohol use of each person's friends
> E<-matrix(0,50,3)
for (i in 1:50)
{
if (sum(s501[i,])!=0)
E[i,1]<-(s501[i,]%*%s50a[,1])/sum(s501[i,])
if (sum(s502[i,])!=0)
E[i,2]<-(s502[i,]%*%s50a[,2])/sum(s502[i,])
if (sum(s503[i,])!=0)
E[i,3]<-(s503[i,]%*%s50a[,3])/sum(s503[i,])
}

## create the dataset to estimate the dynamic linear-in-mean influence model
> alcohol<-c(s50a[,3],s50a[,2])
> lag_alc<-c(s50a[,2],s50a[,1])
> expo<-c(E[,2],E[,1])
> drug<-c(s50d[,3],s50d[,2])
> smoke<-c(s50s[,3],s50s[,2])
> sport<-c(s50sp[,3],s50sp[,2])
> latent_pos2<-rep(m2$mkl$Z,2)
> latent_pos1<-rep(m1$mkl$Z,2)
> infl<-data.frame(cbind(alcohol,lag_alc,expo,drug,smoke,sport,
+ latent_pos1,latent_pos2,rep(c(1:50),2),rep(c(1:2),each=50)))
> head(infl)

   alcohol lag_alc     expo drug  smoke  sport  latent_pos1  latent_pos2 V9  V10
1        3       1 4.333333    1      1      1    -5.997364    -8.472008  1    1
2        2       2 4.000000    3      3      1    -7.324663    -2.941830  2    1
3        3       3 2.500000    1      1      1     6.734962     9.064313  3    1
4        2       3 3.000000    1      1      1     6.734962     9.197778  4    1
5        4       3 3.500000    3      1      2     1.945568     7.413702  5    1
6        4       4 5.000000    1      3      2    18.585402     1.648355  6    1
```

We can also look at the correlations between the estimated latent positions and the observed variables:

```
> cor(infl[,1:8])


            alcohol  lag_alc   expo   drug  smoke   sport  latent_pos1  latent_pos2
alcohol      1.0000    0.699  0.458  0.455  0.386  -0.092     -0.387       -0.317
lag_alc      0.6992    1.000  0.461  0.455  0.465  -0.165     -0.403       -0.364
```

```
expo         0.4585    0.461  1.000  0.348  0.416  -0.221    -0.550       -0.241
drug         0.4553    0.455  0.348  1.000  0.592  -0.382    -0.283       -0.453
smoke        0.3863    0.465  0.416  0.592  1.000  -0.224    -0.340       -0.463
sport       -0.0922   -0.165 -0.221 -0.382 -0.224   1.000     0.145        0.162
latent_pos1 -0.3872   -0.403 -0.550 -0.283 -0.340   0.145     1.000        0.150
latent_pos2 -0.3173   -0.364 -0.241 -0.453 -0.463   0.162     0.150        1.000
```

From the correlation table, strong network autocorrelations are observed – one's alcohol use `alcohol`, previous alcohol use `lag_alc`, and friends' alcohol use expo are all highly correlated with each other. Furthermore, the estimated latent positions in 1995 and 1996 `latent_pos1` and `latent_pos2` have sizable correlations with both girls' alcohol use and their friends' alcohol use. As the calculations of the latent positions are already conditioned on homophily based on the observed variables such as alcohol, drug, smoking, and sport, the results suggest that there might be some unobserved variables (e.g., an unobserved tendency for substance abuse) that drive both girls' alcohol use and choice of friends.

To estimate the dynamic linear-in-mean influence model, I first estimate an influence model with the latent positions as the additional covariates and then estimate another model without the latent positions:

```
> summary(lm(alcohol~lag_alc+expo+smoke+sport+drug+latent_pos1+latent_pos2,data=infl))

Call:
lm(formula = alcohol ~ lag_alc + expo + smoke + sport + drug +
    latent_pos1 + latent_pos2, data = infl)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2031 -0.5060  0.1155  0.5177  1.6341

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.625653   0.453098   1.381   0.1707
lag_alc      0.525024   0.084136   6.240 1.32e-08 ***
expo         0.128865   0.083382   1.545   0.1257
smoke       -0.071843   0.120567  -0.596   0.5527
sport        0.235112   0.168289   1.397   0.1658
drug         0.251790   0.122337   2.058   0.0424 *
latent_pos1 -0.007709   0.011007  -0.700   0.4854
latent_pos2 -0.004525   0.014706  -0.308   0.7590
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7715 on 92 degrees of freedom
Multiple R-squared:  0.5426,	Adjusted R-squared:  0.5078
F-statistic: 15.59 on 7 and 92 DF,  p-value: 2.541e-13

> summary(lm(alcohol~lag_alc+expo+smoke+sport+drug,data=infl))

Call:
lm(formula = alcohol ~ lag_alc + expo + smoke + sport + drug,
    data = infl)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2382 -0.4876  0.0384  0.4935  1.6371

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.47833    0.40080   1.193   0.2357
lag_alc      0.53760    0.08160   6.588 2.54e-09 ***
expo         0.15298    0.07516   2.035   0.0446 *
smoke       -0.05760    0.11602  -0.496   0.6207
sport        0.23565    0.16698   1.411   0.1615
drug         0.26057    0.11873   2.195   0.0307 *
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7655 on 94 degrees of freedom
Multiple R-squared:  0.5398,       Adjusted R-squared:  0.5154
F-statistic: 22.06 on 5 and 94 DF,  p-value: 1.473e-14
```

Results show that if I only include previous alcohol use and other observed covariates, the social influence effect on alcohol use is significant (coef=.152, se=.075, p=.045) – that is, if these girls' friends use more alcohol, they will also use more alcohol. However, when I include the latent positions as the additional covariates in the model, the social influence effect (coef=.129, se=.083, p=.126) is no longer significant.[5] These results suggest that there are likely to be unobserved variables that drive both girls' alcohol use and choice of friends (e.g. unobserved substance-abuse tendency), and ignoring them will lead to 18% overestimation of the social influence effect in this case, which can lead to erroneous statistical inferences.

Next, I estimate a Stochastic Actor-Oriented Model (SAOM) using **RSiena** to test if there is any social influence effect on girls' alcohol use. SAOM is a class of simulation-based statistical models that can model behavioral and network change simultaneously. In the simulation process, SAOM assumes the underlying time is continuous and that actors control their behavior and outgoing ties. At a given moment, one probabilistically selected actor has the opportunity to change one outgoing tie or small step in his or her behavior. The change follows a Markov process in which small changes in networks and behavior are accumulated in each micro-step, and large differences can then be observed between initial and final networks (Snijders et al., 2010). For statistical inference, the parameter values of the simulation algorithms are selected such that the simulated and observed data resemble each other most closely, and the parameters can be estimated by matching key statistics of the simulated and observed networks via the method of moments, generalized method of moments, or likelihood-based methods (Steglich et al., 2010). SAOM is appealing as it intuitively incorporates both the influence and network-selection process from an individual-level perspective, such that the network-selection effects are adjusted for in the estimation of influence effects. However, estimates from SAOMs are still likely to be biased when unobserved variables are co-determining the probabilities of change in network and/or behavior (Steglich et al., 2010) and thus may benefit from the latent space adjusted approach in these scenarios.

I start by constructing a dataset that can be used by SAOM models for estimation:

```
## create data structure that can be used to estimate SAOM
> friend.data.w1 <- s501
> friend.data.w2 <- s502
> friend.data.w3 <- s503
> drink <- s50a
> smoke <- s50s
> drug  <- s50d
> sport <- s50sp
> friendship <- sienaDependent( array( c( friend.data.w1, friend.data.w2,
+                                          friend.data.w3 ),
+                                  dim = c( 50, 50, 3 ) ) )
> drinkingbeh <- sienaDependent( drink, type = "behavior" )
> smokingbeh <- varCovar( as.matrix(smoke))
> drugbeh <- varCovar( as.matrix(drug))
> sportbeh <- varCovar( as.matrix(sport))
> lat1<-coCovar(as.vector(m1$mkl$Z)) ## latent position from 1995
> lat2<-coCovar(as.vector(m2$mkl$Z)) ## latent position from 1996
> myCoEvolutionData <- sienaDataCreate( friendship, drinkingbeh,
+                      smokingbeh,drugbeh,sportbeh,lat1,lat2 )
```

To specify the SAOM model, the following codes can be used. Specifically, in the selection part of the model, I include structural effects such as reciprocity, transitivity, popularity, geometrically weighted degree, and homophily based on alcohol, drug use, smoking, sport, and the latent positions. In the behavioral part of the model, I model girls' alcohol use as a function of the linear and quadratic shapes, average similarity effect (i.e., social influence effect), the observed covariates such as drug use, smoking, sport, as well as the latent positions as the additional covariates:

[5]Latent space model uses a MCMC estimation and thus the results will be slightly different each time. It is suggested to estimate latent space model with longer burn-in, larger sample size, and over multiple times to acquire the final estimates (e.g., using mean or mode of the estimates).

```
> myCoEvolutionEff2 <- getEffects( myCoEvolutionData )
>
> effectsDocumentation(myCoEvolutionEff2)
>
## specify predictors to model selection/network in SAOM
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2, transTrip,
+                                    cycle3,gwespFF,inPop,outPop)
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2, simX,
+                                    interaction1 = "smokingbeh" )
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2, simX,
+                                    interaction1 = "drugbeh" )
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2, simX,
+                                    interaction1 = "sportbeh" )
> myCoEvolutionEff2 <- includeEffects(myCoEvolutionEff2,  simX,
+                                    interaction1 = "drinkingbeh" )
> myCoEvolutionEff2 <- includeEffects(myCoEvolutionEff2,  simX,
+                                    interaction1 = "lat1" )
> myCoEvolutionEff2 <- includeEffects(myCoEvolutionEff2,  simX,
+                                    interaction1 = "lat2" )

## specify predictors to model behavior (alcohol use) in SAOM
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh",
+                                     avSim,
+                                     interaction1 = "friendship" )
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh", effFrom,
+                                     interaction1 = "smokingbeh")
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh", effFrom,
+                                     interaction1 = "drugbeh")
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh", effFrom,
+                                     interaction1 = "sportbeh")
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh", effFrom,
+                                     interaction1 = "lat1")
> myCoEvolutionEff2 <- includeEffects( myCoEvolutionEff2,
+                                     name = "drinkingbeh", effFrom,
+                                     interaction1 = "lat2")
```

To estimate the SAOM model, I type:

```
> betterCoEvAlgorithm <- sienaAlgorithmCreate( projname = 's50CoEv_3',
+                             diagonalize = 0.2, doubleAveraging = 0)
>
>
> (ans2 <- siena07( betterCoEvAlgorithm, data = myCoEvolutionData,
+                  effects = myCoEvolutionEff2))

Estimates, standard errors and convergence t-ratios

                                               Estimate   Standard   Convergence
                                                            Error      t-ratio
Network Dynamics
   1. rate constant friendship rate (period 1)  6.7804  ( 1.9520   )   -0.0247
   2. rate constant friendship rate (period 2)  5.5804  ( 1.5074   )   -0.0446
   3. eval outdegree (density)                 -3.8226  ( 0.4710   )   -0.0268
   4. eval reciprocity                          2.2901  ( 0.4352   )   -0.0036
   5. eval transitive triplets                 -1.1221  ( 0.9042   )   -0.0228
   6. eval 3-cycles                             1.1517  ( 0.5529   )   -0.0201
   7. eval GWESP I -> K -> J (69)               2.7667  ( 1.8913   )   -0.0204
   8. eval indegree - popularity                0.1178  ( 0.1121   )   -0.0313
   9. eval outdegree - popularity              -0.5368  ( 0.1570   )   -0.0286
  10. eval lat1 similarity                      0.6507  ( 0.5186   )   -0.0439
  11. eval lat2 similarity                    7.7015  ( 1.3888   )   -0.0198
  12. eval drinkingbeh similarity             0.6110  ( 0.6676   )    0.0086
  13. eval smokingbeh similarity              0.0755  ( 0.2577   )    0.0176
  14. eval drugbeh similarity                 0.8831  ( 0.5177   )   -0.0197
  15. eval sportbeh similarity                0.2044  ( 0.1843   )    0.0627

Behavior Dynamics
  16. rate rate drinkingbeh (period 1)        1.2506  ( 0.3943   )    0.0629
  17. rate rate drinkingbeh (period 2)        1.7510  ( 0.5416   )    0.0174
  18. eval drinkingbeh linear shape           0.3880  ( 0.1903   )   -0.0095
  19. eval drinkingbeh quadratic shape       -0.1304  ( 0.1459   )   -0.0447
  20. eval drinkingbeh average similarity     3.0265  ( 2.2662   )    0.0059
  21. eval drinkingbeh: effect from lat1     -0.0240  ( 0.0235   )    0.0612
  22. eval drinkingbeh: effect from lat2     -0.0169  ( 0.0324   )    0.0166
  23. eval drinkingbeh: effect from smokingbeh -0.3243  ( 0.3157   )   -0.0706
  24. eval drinkingbeh: effect from drugbeh   0.0538  ( 0.2728   )   -0.0154
  25. eval drinkingbeh: effect from sportbeh   0.3266  ( 0.3720   )    0.0066

Overall maximum convergence ratio:    0.1721


Total of 3944 iteration steps.
```

Results show that there is strong homophily based on the latent positions in the selection process. Furthermore, the estimate for average similarity (i.e., social influence effect) effect is 3.03, and the standard error is 2.27. Next, I compare it with a SAOM model that excludes the latent positions in both selection and behavioral models. Results are shown below:

```
Estimates, standard errors and convergence t-ratios

                                              Estimate   Standard   Convergence
                                                           Error      t-ratio
Network Dynamics
   1. rate constant friendship rate (period 1)  5.6744  ( 1.4262   )    0.0123
   2. rate constant friendship rate (period 2)  4.4861  ( 0.9524   )   -0.0206
   3. eval outdegree (density)               -2.3732  ( 0.2822   )    0.0193
   4. eval reciprocity                        3.0429  ( 0.4632   )    0.0205
   5. eval transitive triplets               -1.4128  ( 0.8951   )    0.0193
   6. eval 3-cycles                           1.7027  ( 0.5465   )    0.0205
   7. eval GWESP I -> K -> J (69)             3.6722  ( 1.7601   )    0.0104
   8. eval indegree - popularity              0.0872  ( 0.1019   )   -0.0118
   9. eval outdegree - popularity            -0.6361  ( 0.1700   )    0.0215
  10. eval drinkingbeh similarity             1.2178  ( 0.7357   )    0.0277
  11. eval smokingbeh similarity             -0.0006  ( 0.2812   )   -0.0166
  12. eval drugbeh similarity                 0.9889  ( 0.4224   )   -0.0231
  13. eval sportbeh similarity                0.1628  ( 0.1859   )   -0.0149

Behavior Dynamics
  14. rate rate drinkingbeh (period 1)        1.2869  ( 0.3117   )    0.0219
  15. rate rate drinkingbeh (period 2)        1.7214  ( 0.4520   )    0.0173
  16. eval drinkingbeh linear shape           0.3975  ( 0.1840   )   -0.0159
  17. eval drinkingbeh quadratic shape       -0.0542  ( 0.1209   )    0.0014
  18. eval drinkingbeh average similarity     4.0685  ( 2.0968   )    0.0147
  19. eval drinkingbeh: effect from smokingbeh -0.2452  ( 0.3031   )    0.0476
  20. eval drinkingbeh: effect from drugbeh   0.0836  ( 0.2829   )    0.0277
  21. eval drinkingbeh: effect from sportbeh   0.3029  ( 0.3710   )   -0.0065

Overall maximum convergence ratio:    0.1545


Total of 3743 iteration steps.
```

The estimate for the social influence effect is now 4.07, with a standard error of 2.10. As a result, ignoring the latent position will likely lead to 34% overestimation of the social influence effect in this case using the SAOM models.

Discussion and Conclusion

Social influence effects are generally difficult to identify, as influence processes are often entangled with other processes such as selection and social-environmental factors. Here I have shown that this entanglement/difficulty can essentially be framed as an omitted variable bias problem, and a latent space adjusted approach holds promise to correctly identify social influence effects in this case. And I have demonstrated how to use the latent space adjusted approach to estimate various social influence models with existing packages in R. Results show that models that ignore the unobserved variables that drive both influence and selection are likely to overestimate the true social influence effect, while the latent space adjusted approach holds promise to correct that bias and serves as a more conservative test of the true social influence effect.

Although the latent space adjusted approach proposed in this paper is flexible enough to be incorporated with any functional form of the behavioral/influence model, and holds much promise as an alternative approach to identify the social influence effect, several limitations also come with this approach: (1) As previously mentioned, the latent space adjusted approach requires that the same unobserved traits occur in both the influence and the selection process. It can not account for the unobserved traits that are only present in one of the processes but not the other. (2) The choice of the dimensions for the latent positions in the latent space model is not clear. Although I have chosen one-dimensional latent positions in all of the simulations and empirical examples, this does not need to be the case and there is no clear rule deciding how many dimensions users should use. (3) The computation of latent positions is very time-consuming, and the computation time increases significantly with the increase of data or the number of dimensions of the latent positions.

Nevertheless, the latent space adjusted approach proposed here provides a useful and more plausible estimate of the true social influence effect, especially when the entanglement between influence and selection is of concern. This paper contributes to the literature by further illustrating how the latent space adjusted approach may account for bias in the estimation of the social influence effect, as well as how this approach can be easily implemented in R.

Bibliography

S. Aral, L. Muchnik, and A. Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proceedings of the National Academy of Sciences*, 106(51): 21544–21549, 2009. URL `https://doi.org/10.1073/pnas.0908800106`. [p57]

S. Asch. Group forces in the modification and distortion of judgments. *Social psychology.*, pages 450–501, 1972. URL `https://doi.org/10.1037/10025-016`. [p57]

A. Bandura. *Social foundations of thought and action*. Englewood Cliffs, NJ, 1986. [p57]

Y. Bramoulle, H. Djebbari, and B. Fortin. Identification of peer effects through social networks. *Journal of Econometrics*, 150(1):41–55, 2009. URL `https://doi.org/10.1016/j.jeconom.2008.12.021`. [p57]

J. T. Cacioppo, J. H. Fowler, and N. A. Christakis. Alone in the crowd: The structure and spread of loneliness in a large social network. *SSRN Electronic Journal*, 2008. URL `https://doi.org/10.2139/ssrn.1319108`. [p57]

N. A. Christakis and J. H. Fowler. The spread of obesity in a large social network over 32 years. *New England Journal of Medicine*, 357(4):370–379, 2007. URL `https://doi.org/10.1056/nejmsa066082`. [p57]

N. A. Christakis and J. H. Fowler. The collective dynamics of smoking in a large social network. *New England Journal of Medicine*, 358(21):2249–2258, 2008. URL `https://doi.org/10.1056/nejmsa0706154`. [p57]

Y. Croissant, G. Millo, and K. Tappe. plm: Linear models for panel data, 2021. URL `https://CRAN.R-project.org/package=plm`. R package version 2.4-1. [p60]

L. Erbring and A. Young. Individuals and social structure. *Sociological Methods and Research*, 7(4): 396–430, 1979. URL `https://doi.org/10.1177/004912417900700404`. [p57]

S. L. Feld. The focused organization of social ties. *American Journal of Sociology*, 86(5):1015–1035, 1981. URL `https://doi.org/10.1086/227352`. [p57]

S. L. Feld. Social structural determinants of similarity among associates. *American Sociological Review*, 47(6):797, 1982. URL `https://doi.org/10.2307/2095216`. [p57]

K. A. Frank, Y. Zhao, and K. Borman. Social capital and the diffusion of innovations within organizations: The case of computer technology in schools. *Sociology of Education*, 77(2):148–171, 2004. URL https://doi.org/10.1177/003804070407700203. [p57]

N. E. Friedkin. Norm formation in social influence networks. *Social Networks*, 23(3):167–189, 2001. URL https://doi.org/10.1016/s0378-8733(01)00036-3. [p57]

N. E. Friedkin and E. C. Johnsen. Social influence and opinions. *The Journal of Mathematical Sociology*, 15(3-4):193–206, 1990. URL https://doi.org/10.1080/0022250x.1990.9990069. [p61, 62]

N. E. Friedkin and E. C. Johnsen. Social influence networks and opinion change. *Advances in Group Processes*, 16(1):1–29, 1999. [p57]

D. German, C. G. Sutcliffe, B. Sirirojn, S. G. Sherman, C. A. Latkin, A. Aramrattana, and D. D. Celentano. Unanticipated effect of a randomized peer network intervention on depressive symptoms among young methamphetamine users in thailand. *Journal of Community Psychology*, 40(7):799–813, Jul 2012. URL https://doi.org/10.1002/jcop.21488. [p57]

M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, P. N. Krivitsky, S. Bender-deMoll, and M. Morris. statnet: Software tools for the statistical analysis of network data, 2019. URL https://CRAN.R-project.org/package=statnet. R package version 2019.6. [p60]

P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002. URL https://doi.org/10.1198/016214502388618906. [p57, 59, 60]

M. Kalmijn and H. Flap. Assortative meeting and mating: Unintended consequences of organized settings for partner choices. *Social Forces*, 79(4):1289–1312, Jan 2001. URL https://doi.org/10.1353/sof.2001.0044. [p57]

P. N. Krivitsky and M. S. Handcock. latentnet: Latent position and cluster models for statistical networks, 2020. URL https://CRAN.R-project.org/package=latentnet. R package version 2.10.1. [p60]

C. F. Manski. Identification of endogenous social effects: The reflection problem. *The Review of Economic Studies*, 60(3):531, 1993. URL https://doi.org/10.2307/2298123. [p57]

J. M. Mcpherson and L. Smith-Lovin. Homophily in voluntary organizations: Status distance and the composition of face-to-face groups. *American Sociological Review*, 52(3):370, 1987. URL https://doi.org/10.2307/2095356. [p57]

M. Mcpherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001. URL https://doi.org/10.1146/annurev.soc.27.1.415. [p57]

M. P. L. Michell. Smoke rings: social network analysis of friendship groups, smoking and drug-taking. *Drugs: Education, Prevention and Policy*, 7(1):21–37, 2000. URL https://doi.org/10.1080/dep.7.1.21.37. [p60]

E. R. Oetting and J. F. Donnermeyer. Primary socialization theory: The etiology of drug use and deviance. i. *Substance Use & Misuse*, 33(4):995–1026, 1998. URL https://doi.org/10.3109/10826089809056252. [p57]

M. Pearson and P. West. Drifting smoke rings. *Connections*, 25(2):59–76, 2003. [p60]

R. Ripley, K. Boitmanis, T. A. Snijders, and F. Schoenenberger. Rsiena: Siena - simulation investigation for empirical network analysis, 2018. URL https://CRAN.R-project.org/package=RSiena. R package version 1.2-12. [p60, 61]

Y. Rosseel and T. D. Jorgensen. lavaan: Latent variable analysis, 2019. URL https://CRAN.R-project.org/package=lavaan. R package version 0.6-5. [p60]

C. R. Shalizi and E. McFowland III. Estimating causal peer influence in homophilous social networks by inferring latent locations. *arXiv*, 33, 2018. URL arXiv:1607.06565. [p57, 60]

C. R. Shalizi and A. C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods & Research*, 40(2):211–239, 2011. URL https://doi.org/10.1177/0049124111404820. [p57, 58]

S. Shortreed, M. S. Handcock, and P. Hoff. Positional estimation within a latent space model for networks. *Methodology*, 2(1):24–33, 2006. URL https://doi.org/10.1027/1614-2241.2.1.24. [p59]

T. A. Snijders, G. G. V. D. Bunt, and C. E. Steglich. Introduction to stochastic actor-based models for network dynamics. *Social Networks*, 32(1):44–60, 2010. URL https://doi.org/10.1016/j.socnet.2009.02.004. [p57, 61, 64]

C. Steglich, T. A. B. Snijders, and M. Pearson. 8. dynamic networks and behavior: Separating selection from influence. *Sociological Methodology*, 40(1):329–393, 2010. URL https://doi.org/10.1111/j.1467-9531.2010.01225.x. [p58, 64]

G. Szekely, M. L. Rizzo, and N. K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007. URL https://doi.org/10.1214/009053607000000505. [p59]

T. W. Valente. Network models and methods for studying the diffusion of innovations. *Models and Methods in Social Network Analysis*, pages 98–116, 1995. URL https://doi.org/10.1017/cbo9780511811395.006. [p57]

T. W. Valente. Social network thresholds in the diffusion of innovations. *Social Networks*, 18(1):69–89, 1996. URL https://doi.org/10.1016/0378-8733(95)00256-1. [p57]

T. J. Vanderweele and W. An. Social networks and causal inference. *Handbooks of Sociology and Social Research Handbook of Causal Analysis for Social Research*, pages 353–374, 2013. URL https://doi.org/10.1007/978-94-007-6094-3_17. [p58]

J. M. Wooldridge. *Econometric analysis of cross section and panel data*. MIT Press, 2011. [p59]

R. Xu. Alternative estimation methods for identifying contagion effects in dynamic social networks: A latent-space adjusted approach. *Social Networks*, 54:101–117, 2018. URL https://doi.org/10.1016/j.socnet.2018.01.002. [p57, 58, 59, 60]

R. Xu. Statistical methods for the estimation of contagion effects in human disease and health networks. *Computational and Structural Biotechnology Journal*, 18:1754–1760, 2020. URL https://doi.org/10.1016/j.csbj.2020.06.027. [p58]

Ran Xu
Department of Allied Health Sciences, University of Connecticut
Storrs, CT. 06269
USA
(ORCiD:0000-0002-5832-9226)
ran.2.xu@uconn.edu

survidm: An R package for Inference and Prediction in an Illness-Death Model

by Gustavo Soutinho, Marta Sestelo and Luís Meira-Machado

Abstract Multi-state models are a useful way of describing a process in which an individual moves through a number of finite states in continuous time. The illness-death model plays a central role in the theory and practice of these models, describing the dynamics of healthy subjects who may move to an intermediate "diseased" state before entering into a terminal absorbing state. In these models, one important goal is the modeling of transition rates which is usually done by studying the relationship between covariates and disease evolution. However, biomedical researchers are also interested in reporting other interpretable results in a simple and summarized manner. These include estimates of predictive probabilities, such as the transition probabilities, occupation probabilities, cumulative incidence functions, and the sojourn time distributions. The development of **survidm** package has been motivated by recent contribution that provides answers to all these topics. An illustration of the software usage is included using real data.

Introduction

Multi-state models are very useful for describing complex event history data with multiple endpoints. These models may be considered a generalization of survival analysis where survival is the ultimate outcome of interest but where information is available about intermediate events which individuals may experience during the study period. For instance, in most biomedical applications, besides the 'healthy' initial state and the absorbing 'dead' state, one may observe intermediate (transient) states based on health conditions (e.g., diseased), disease stages (e.g., stages of cancer or HIV infection), clinical symptoms (e.g., bleeding episodes), biological markers (e.g., CD4 T-lymphocyte cell counts; serum immunoglobulin levels), or they can represent a non-fatal complication in the course of the illness (e.g., cancer recurrence, transplantation, etc.). Graphically, these models may be illustrated using diagrams with boxes representing the states and with arrows between the states representing the possible transitions. The complexity of the multi-state model greatly depends on the number of states and also on the possible transitions. The illness-death model is probably the most popular one in the medical literature. The irreversible version of this model (Figure 1) describes the pathway from an initial state to an absorbing state either directly or through an intermediate state. Many event-history data sets from biomedical studies with multiple endpoints can be reduced to this generic structure. There exists extensive literature on multi-state models. Main contributions include books by Andersen et al. (1993) and Hougaard (2000) (Chapter 5 and 6). Recent reviews on this topic may be found in the papers by Putter et al. (2007), Meira-Machado et al. (2009), and Meira-Machado and Sestelo (2019).

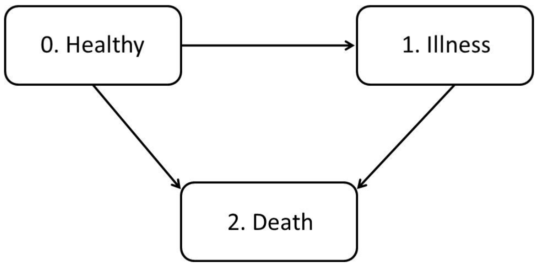


Figure 1: Progressive illness-death model.

One important goal in multi-state modeling is to relate the individual characteristics with the intensity rates through a covariate vector, but biomedical researchers are also interested in reporting interpretable results in a simple and summarized manner. These include estimates of predictive probabilities, such as the transition probabilities, occupation probabilities, cumulative incidence functions, and the sojourn time distributions. The development of **survidm** R package has been motivated by several recent contributions that account for these problems; in particular the newly developed methods based on subsampling (see Meira-Machado and Sestelo (2019) for further details). The current version of the package provides seven different approaches to estimate the transition probabilities, two methods for the sojourn distributions and two approaches for the cumulative incidence functions. In addition, these probabilities can also be estimated conditionally on covariate measures. The package also allows the user to perform multi-state regression where the estimation of the covariate effects is achieved using Cox regression in which different effects of the covariates are assumed for different transitions.

Several researchers have recently developed software for multi-state survival analysis. A comprehensive list of the available packages in the Comprehensive R Archive Network (CRAN) can be seen in the CRAN task view 'Survival Analysis' (Allignol and Latouche, 2018). In R, several packages provide functions for estimating the transition probabilities (e.g., the package **p3state.msm** (Meira-Machado and Roca-Pardiñas, 2011), **TPmsm** (Araújo et al., 2014), **etm** (Allignol et al., 2011), **mstate** (de Wreede et al., 2011), and **TP.idm** (Balboa and de Uña-Álvarez, 2018)), but none implements all the methods addressed by **survidm** which includes all newly developed methods based on the subsampling approach (see de Uña-Álvarez and Meira-Machado (2015) and references therein). In addition, not all allow the users to obtain estimates of the transition probabilities conditional to covariates. The **cmprsk** and the **timereg** R packages can be used to estimate the cumulative incidence functions in a competing risks model. The package survival (via `survfit` and `coxph` functions) can also be used for competing risks data. The **msSurv** can be used to estimate the state occupation probabilities and the sojourn distributions for multi-state models subject to right-censoring (possibly state-dependent) and left-truncation. The package also provides matrices of transition probabilities between any two states. However, none of the available software provides an encompassing package which can be used to estimate all these quantities. Finally, the use of different packages for estimating these quantities separately is rather difficult because each of the current programs requests its own data structure. This paper introduces **survidm** (available from the Comprehensive R Archive Network at `https://cran.r-project.org/web/packages/survidm/`), a software application for R which performs inference in a progressive illness-death model. It describes the capabilities of the program for estimating semiparametric regression models and for implementing nonparametric estimators for all quantities mentioned above.

The remainder of this paper is organized as follows. The following section provides a brief introduction to the methodological background. Then, a detailed description of the package is presented, and its usage is illustrated through the analysis of a real data set. Finally, the last section contains the main conclusions of this work.

Methodology background

The mathematical background underlying the **survidm** package is briefly introduced in this section. A more detailed introduction can be found in Meira-Machado and Sestelo (2019). The present contribution builds on this article by offering guidelines for using the software to implement the proposed methods.

Notation

A multi-state model is a model for a time-continuous stochastic process $(Y(t), t \geq 0)$ which at any time occupies one of a few possible states. In this paper, we consider the progressive illness-death model depicted in Figure 1, and we assume that states are numbered as $0 - \textit{healthy}$, $1 - \textit{illness}$, and $2 - \textit{death}$. We also assume that all subjects enter the study in State 0 and that they may either visit State 1 at some time point; or not, going directly to the absorbing state (State 2).

This model is characterized by the joint distribution of (Z, T), where Z denotes the sojourn time in the initial State 0, and T is the total survival time of the process. As usual with survival data, individuals are generally followed over a certain period of time, providing right-censored observations which are modeled by considering a censoring variable C, which we assume to be independent of of (Z, T). Due to censoring, rather than (Z, T), we observe $\widetilde{Z} = \min(Z, C)$, $\widetilde{T} = \min(T, C)$, $\Delta_1 = I(Z \leq C)$, and $\Delta = I(T \leq C)$ for the respective censoring indicators of Z and T. Finally, the available data is $(\widetilde{Z}_i, \widetilde{T}_i, \Delta_{1i}, \Delta_i)$, $1 \leq i \leq n$, i.i.d. copies of $(\widetilde{Z}, \widetilde{T}, \Delta_1, \Delta)$.

Regression models for transitions intensities

One important goal in multi-state modeling is to study the relationships between the different predictors and the outcome. To relate the individual characteristics to the intensity rates, several models have been used in the literature. A common simplifying strategy is to decouple the whole process into various survival models by fitting separate intensities to all permitted transitions using semiparametric Cox proportional hazard regression models (Cox, 1972), while making appropriate adjustments to the risk set. The most common models are characterized through one of the two model assumptions that can be made about the dependence of the transition intensities and time. The transition intensities may be modeled using separated Cox models assuming the process to be Markovian (also known as the clock forward modeling approach), which states that past and future are independent given the present state. They can also be modeled using a semi-Markov model in which the future of the process does not depend on the current time but rather on the duration in the current state. Semi-Markov models are also called 'clock reset' models because each time the patient enters a new state, the time is reset to 0. The package **survidm** is restricted to these two semiparametric multi-state models, but other models are possible for the analysis of multi-state survival data. For example, time-homogeneous markov models and model with piecewise constant intensities are implemented in the **msm** R package (Jackson, 2011). Aalen additive model (Aalen et al., 2001) and accelerated failure time models (Wei, 1992) are another class of regression models that can be an alternative to the Cox proportional hazards model.

Transition probabilities

For two states h, j and two time points $s < t$, the so-called transition probabilities $p_{hj}(s,t) = P(Y(t) = j|Y(s) = h)$ are introduced. In the progressive illness-death model, there are five different transition probabilities to estimate: $p_{00}(s,t)$, $p_{01}(s,t)$, $p_{02}(s,t)$, $p_{11}(s,t)$, and $p_{12}(s,t)$. Since $p_{00}(s,t) + p_{01}(s,t) + p_{02}(s,t) = 1$ and $p_{11}(s,t) + p_{12}(s,t) = 1$, in practice, one only needs to estimate three of these quantities. The state occupation probabilities are defined as $p_j(t) = P(Y(t) = j)$. If we assume that all subjects are in State 0 at time $t = 0$, then $p_j(t) = p_{0j}(0,t)$ and, therefore, the occupation probabilities can be seen as a particular case of the transition probabilities. Estimating these quantities is interesting since they allow for long-term predictions of the process.

The standard nonparametric method to estimate a transition probability matrix is the time-honored Aalen-Johansen (AJ) estimator (Aalen and Johansen, 1978). This estimator benefits from the assumption of Markovianity on the underlying stochastic process extending the time-honored Kaplan-Meier estimator (Kaplan and Meier, 1958) to Markov chains. Explicit formulae of the Aalen-Johansen estimator for the illness-death model are available (Borgan, 1988).

Moreira et al. (2013) propose a modification of the Aalen-Johansen estimator in the illness-death model based on a preliminary smoothing (also known as presmoothing, Dikta (1998); Cao et al. (2005)) of the censoring probability for the total time (respectively, of the sojourn time in State 0), given the available information. The presmoothed Aalen-Johansen (PAJ) estimator proposed by Moreira et al. (2013) is obtained by replacing the censoring indicators (in the transition probabilities $p_{00}(s,t)$ and $p_{11}(s,t)$) by an estimator of a binary (logistic) regression function. The authors verified through simulations that the use of presmoothing can lead to improved estimators with less variability.

The Markov assumption may be violated in practice. For example, for the progressive illness-death model, the arrival time to the intermediate state of the process often influences the subsequent transition hazard, leading to non-Markov structures. If the Markov property is violated, then the consistency of the time-honored Aalen-Johansen estimator and of its presmoothed versions can not be ensured in general. Exceptions to this are the estimators for $p_{00}(s,t)$ or for the so-called occupation probabilities, $p_{0j}(0,t)$ (Datta and Satten, 2001).

Estimators for the transition probabilities in the progressive illness-death model, which do not rely on the Markov assumption, were introduced for the first time by Meira-Machado et al. (2006). The proposed estimators were defined in terms of multivariate Kaplan-Meier integrals with respect to the marginal distributions of Z and T. These authors showed the practical superiority of their estimators relative to the Aalen-Johansen in situations in which the Markov condition is strongly violated. However, their proposal has the drawback of requiring that the support of the censoring distribution contains the support of the lifetime distribution. Otherwise, they only report valid estimators for truncated transition probabilities. To avoid this issue, corrected estimators (labeled in this paper as LIDA, the acronym of Lifetime Data Analysis, the journal in which this estimator was published for the first time) were proposed by de Uña-Álvarez and Meira-Machado (2015) for $p_{01}(s,t)$ and $p_{11}(s,t)$.

The paper by de Uña-Álvarez and Meira-Machado (2015) also introduces estimators based on subsampling. The idea behind subsampling, also referred to as landmarking (Van Houwelingen, 2007), is to consider the subset of individuals observed in State h by time s. To be specific, given the time point s, to estimate $p_{0j}(s,t)$ for $j = 0,1,2$, the landmark analysis is restricted to the individuals observed in State 0 at time s. Whereas, to estimate $p_{1j}(s,t)$, $j = 1,2$, the landmark analysis proceeds from the sample restricted to the individuals observed in State 1 at time s. The procedure is then based on (differences between) Kaplan-Meier estimators derived from these subsets of the data. These estimators are termed LM in the present paper as well as in the **survidm** package.

In some cases, subsampling leads to small sample sizes which may result in estimators with high variability. To avoid this problem, a valid approach is to consider a modification of the landmark estimator based on presmoothing (Meira-Machado, 2016). The presmoothed landmark estimators (PLM) are a good alternative in these situations since they give mass to all the event times, including the censored observations.

Subsampling was later used by Putter and Spitoni (2018) to derive a landmark Aalen-Johansen estimator (`LMAJ`) of the transition probabilities. The idea behind the proposed estimator is to use the Aalen-Johansen estimator of the state occupation probabilities derived from those subsets (consisting of subjects occupying a given state at a particular time) for which consistency have already been proved in multi-state models that are not necessarily Markov (Datta and Satten, 2001). In this latter approach, the application of presmoothed estimators (`PLMAJ`) is possible too.

Also of interest is the estimation of the transition probabilities given a covariate (or a vector of covariates) that is observed for an individual before the individual makes a particular transition of interest. One standard method, particularly well-suited to the setting with multiple covariates, is to consider estimators based on a Cox's regression model (Cox, 1972) fitted marginally to each transition with the corresponding baseline hazard function estimated by the Breslow's method (Breslow, 1972). One alternative and flexible nonparametric approach is to consider local smoothing by means of kernel weights based on local constant (Nadaraya-Watson) regression. Right censoring is handled by applying inverse probability of censoring weighting. This is a fully nonparametric approach which provides flexible effects of the continuous covariates (Meira-Machado et al., 2015; Rodríguez-Álvarez et al., 2016; Meira-Machado and Sestelo, 2019). The two possible approaches are implemented in the **survidm** package and labeled as `breslow` and `IPCW`, respectively.

Cumulative incidence functions

Another quantity of interest in multi-state modeling is the cause-specific cumulative incidence function, as defined by Kalbfleisch and Prentice (1980). In the illness-death model, two cumulative incidence functions are of particular interest: the cumulative incidence of the illness and the cumulative incidence of dying without the disease. These quantity represents the probability of an individual being or having been diseased at time t. One possible estimator for the cause-specific cumulative incidence function in a competing risks setting can be performed using the estimator proposed by Geskus (2011). This estimator based on the subdistribution hazard is obtained by applying the Nelson-Aalen estimator and the product-limit estimator of the disease-free survival. This estimator can also be expressed in terms of the Kaplan-Meier weights of the distribution of Z, the sojourn time in State 0, as introduced in the paper by Meira-Machado and Sestelo (2019). A modification of this estimator based on presmoothing can be introduced to reduce its variability. Both methods are implemented in the **survidm** package. Estimation methods for the cumulative incidence function conditionally on covariate measures based on local constant (Nadaraya-Watson) regression are also implemented in the package.

Sojourn distributions

The estimation of the marginal distributions in multi-state modeling is an interesting topic too. In the context of the illness-death model, if the independence assumption between the censoring variable C and the vector of times (Z, T) is assumed, the marginal distribution of the sojourn time in State 0, Z, can be consistently estimated by the Kaplan-Meier estimator based on the $(\widetilde{Z}_i, \Delta_{1i})$'s. Similarly, the distribution of the total time may be consistently estimated by the Kaplan-Meier estimator based on the $(\widetilde{T}_i, \Delta_i)$'s. However, the estimation of the marginal distribution of the sojourn time in State 1 is not such a simple issue. Nonparametric estimates for this marginal distribution allowing for state and path-dependent censoring were proposed by Satten and Datta (2002).

survidm in practice

This section introduces an overview of how the package is structured.

This software enables both numerical and graphical outputs to be displayed for all methods described in the previous section. This software is intended to be used with the R statistical program (R Core Team, 2019). Our package is composed of 17 functions that allow users to obtain estimates for all proposed methods. Details on the usage of the functions (described in Table 1) can be obtained with the corresponding help pages.

It should be noted that to implement the methods described in the methodology section, one needs the following variables of data: `time1`, `event1`, `Stime`, and `event`. Covariates can also be included. The variable `time1` represents the sojourn time in State 0 and `Stime` the total time, whereas `event1` and `event` are the respective censoring indicators. This means that `event1` will take the value 1 if the subject leaves State 0 and 0 otherwise; `event` takes value 1 if the subject reaches State 2 and 0 otherwise.

Function	Description
`survIDM`	Create a `survIDM` object.
`coxidm`	Fits proportional hazards regression models for each transition.
`tprob`	Estimation of the transition probabilities.
`CIF`	Estimation of the cumulative incidence functions.
`sojourn`	Nonparametric estimation of the sojourn distribution in the intermediate state.
`autoplot.survIDM`	Visualization of `survIDM` objects with **ggplot2** and **plotly** graphics.
`plot.survIDM`	Plot for an object of class `survIDM`.
`print.survIDM`	Print for an object of class `survIDM`.
`summary.survIDM`	Summary for an object of class `survIDM`.
`nevents`	Counts the number of observed transitions in the multi-state model.
`markov.test`	Performs a test for the Markov assumption.
`KM`	Computes the Kaplan-Meier product-limit of survival.
`PKM`	Computes the presmoothed Kaplan-Meier product-limit of survival.
`Beran`	Computes the conditional survival probability of the response, given the covariate under random censoring.
`KMW`	Returns a vector with the Kaplan-Meier weights.
`PKMW`	Returns a vector with the presmoothed Kaplan-Meier weights.
`LLW`	Returns a vector with the local linear weights.
`NWW`	Returns a vector with the Nadaraya-Watson weights.

Table 1: Summary of functions in the **survidm** package.

For illustration, we apply the proposed methods to data from a large clinical trial on Duke's stage III patients affected by colon cancer that underwent a curative surgery for colorectal cancer (Moertel et al., 1990). This data set is freely available as part of the R **survival** package. The data is also available as part of the R package **survidm**. Besides the two event times (disease-free survival time and death time) and the corresponding indicator statuses, a vector of covariates including rx (treatment: Obs(ervation), Lev(amisole), Lev(amisole)+5FU), sex (1 - male), age (years), nodes (number of lymph nodes with detectable cancer), surge (time from surgery to registration: 0 = short, 1 = long), adhere (adherence to nearby organs) are also available. The covariate 'recurrence' is the only time-dependent covariate, while the other covariates included are fixed. Recurrence can be considered as an intermediate transient state and modeled using the progressive illness-death model with transient states 'alive and disease-free' and 'alive with recurrence', and the absorbing state 'dead'. In the following, we will demonstrate the package capabilities using this data. Below is an excerpt of the data.frame with one row per individual. Individuals were chosen in order to represent all possible combinations of movements among the three states.

```
> library("survidm")
> data(colonIDM)
> colonIDM[c(1:2,16,21),1:7]

   time1 event1 Stime event      rx sex age
1    968      1  1521     1 Lev+5FU   1  43
2   3087      0  3087     0 Lev+5FU   1  63
16  1323      1  3214     0     Obs   1  68
21  2789      1  2789     1     Obs   1  64
```

Individual represented in the first line experienced a recurrence of the tumor and have died. In such cases, `event1` = 1 and `time1` = `Stime` indicate that the individual observed a direct transition from State 0 to State 1 (with `event1` = 1). Individual represented in line 2 remain alive and without recurrence at the end of follow-up (`event1` = 0 and `event` = 0). Individual represented in line 16 of the original data set, with `event1 = 1` and `event = 0`, corresponds to an individual with an observed recurrence that remains alive at the end of the follow-up. Note that in this case, the disease-free survival time is equal to the death time (`time1` = `Stime`). Finally, individual represented in line 21 of the original data set has died without observing a recurrence. We note that `event1 = 1` and `event = 0` correspond to individuals with an observed recurrence that remain alive at the end of the follow-up.

Of the total of 929 patients, 468 developed a recurrence, and among these 414 died, 38 patients died without developing a recurrence. A summary of the data with the number of the undergoing transitions can be obtained through the `nevents` function. The colums of the data set must include at least the four columns named `time1`, `event1`, `Stime`, and `event` according to the requirements of the `survIDM` function presented in the help file. Parameter `state.names` enables to change the default values of states, 'healthy', 'illness', and 'death'.

```
> nevents(with(colonIDM, survIDM(time1, event1, Stime, event)),
          state.names = c("healthy", "recurrence", "death"))

           healthy recurrence death
healthy        423        468    38
recurrence       0         54   414
death            0          0   452
```

Regression models for transitions intensities

To relate the individual characteristics to the intensity rates, semiparametric multi-state regression models are used. Specifically, separated Cox models assuming the process to be Markovian (i.e., the transition intensities only depend on the history of the process through the current state) or using a semi-Markov model in which the future of the process does not depend on the current time but rather on the duration in the current state. Therefore, practical interest to determine whether the Markov property holds within a particular data set to determine whether a Markov model or a semi-Markov model is more appropriate.

The Markov assumption

The Markov assumption may be checked, among others, by including covariates depending on the history. For the progressive illness-death model, the Markov assumption is only relevant for mortality transition after recurrence. We can examine whether the time spent in the initial state "Alive and disease-free" (i.e., the past) is important in the transition from the recurrence state to death (i.e., the future). For doing that, let Z be the time spent in State 0 and t the current time. Fitting a model $\alpha_{12}(t;Z) = \alpha_{12,0}(t)exp\{\beta Z\}$, we now need to test the null hypothesis, $H_0 : \beta = 0$, against the general alternative, $H_1 : \beta \neq 0$. This would assess the assumption that the transition rate from the disease state into death is unaffected by the time spent in the previous state.

```
> library(survival)
> fit <- coxph(Surv(time1, Stime, event) ~ time1, data = colonIDM,
               subset=c(time1 < Stime))
> fit
           coef  exp(coef)   se(coef)      z     p
time1 -0.0002475  0.9997526  0.0001737 -1.424 0.154

Likelihood ratio test=2.04  on 1 df, p=0.1533
n= 468, number of events= 414
```

Following this procedure, we verified that the effect of time spent in State 0 reported a p-value of 0.154 (regression coefficient: - 0.0002475), revealing no evidence against the Markov model for the colon data. Results from this test can also be obtained through the function `markov.test`, which has an output fairly similar to those obtained from `coxph` function.

```
> mk <- markov.test(survIDM(time1, event1, Stime, event) ~ 1, data = colonIDM)
> mk
```

Since there is no evidence on the lack of Markovianity, a multi-state Markov regression model based on the Cox model can be fitted through the following input command:

```
> fit.cmm <- coxidm(survIDM(time1, event1, Stime, event) ~ rx + sex + age +
                     nodes + surg + adhere, data = colonIDM)

> summary(fit.cmm)

Cox Markov Model: transition 0 -> 1

                  coef exp(coef) lower 0.95 upper 0.95     Pr(>|z|)
rxLev     -0.061251858 0.9405863  0.7596976  1.1645457 5.740592e-01
rxLev+5FU -0.515170844 0.5973985  0.4713678  0.7571264 2.031682e-05
sex       -0.149177218 0.8614164  0.7160077  1.0363552 1.137849e-01
```

```
age       -0.004669254 0.9953416  0.9876802  1.0030625 2.362711e-01
nodes      0.083943790 1.0875678  1.0686993  1.1067694 5.418662e-21
surg       0.251798521 1.2863368  1.0509673  1.5744186 1.460249e-02
adhere     0.296839791 1.3455997  1.0551768  1.7159575 1.671466e-02


Cox Markov Model: transition 0 -> 2

                 coef exp(coef) lower 0.95 upper 0.95     Pr(>|z|)
rxLev     -0.29152482 0.7471235  0.3271685   1.706135 4.889711e-01
rxLev+5FU -0.11211853 0.8939383  0.4220165   1.893589 7.697006e-01
sex        0.39293182 1.4813174  0.7641923   2.871399 2.445966e-01
age        0.08422764 1.0878765  1.0476871   1.129608 1.157046e-05
nodes      0.07538428 1.0782984  0.9895116   1.175052 8.552937e-02
surg       0.41564547 1.5153485  0.7703441   2.980851 2.285509e-01
adhere     0.05435239 1.0558566  0.4377875   2.546517 9.036879e-01


Cox Markov Model: transition 1 -> 2

                  coef exp(coef) lower 0.95 upper 0.95     Pr(>|z|)
rxLev      0.068953592  1.071386  0.8533466   1.345138 5.525534e-01
rxLev+5FU  0.327043851  1.386862  1.0741245   1.790656 1.212756e-02
sex        0.214094887  1.238740  1.0138220   1.513557 3.623833e-02
age        0.009342474  1.009386  1.0014760   1.017359 1.994502e-02
nodes      0.046061552  1.047139  1.0249376   1.069821 2.522475e-05
surg      -0.012258877  0.987816  0.7944594   1.228232 9.121722e-01
adhere     0.137708158  1.147641  0.8851963   1.487895 2.985854e-01
```

The transition intensities characterize the hazard for movement from one state to another, revealing how the different covariates affect the various permitted transitions. The results obtained indicate that none of the covariates were found to have a strong effect on all three transitions. Save for covariates age and sex, all the remaining predictors were considered important for recurrence transition. Interestingly, age displayed a strong linear effect on mortality transition without recurrence, whereas all the other covariates failed to show relevant association on this transition. Finally, save for covariates surg and adhere, all the remaining predictors were considered important for the mortality transition after recurrence. The `coxidm` function also returns the analysis of the deviance for each Cox model. In this case, only an overall p-value is presented for categorical variables. To obtain the outputs, we have to indicate `type='anova'` in `summary` function.

```
> summary(fit.cmm,type = 'anova')

Cox Markov Model: transition 0 -> 1

        loglik   Chisq Df Pr(>|Chi|)
NULL   -2954.2
rx     -2941.8 24.6964  2  4.338e-06 ***
sex    -2941.0  1.6402  1    0.20030
age    -2939.8  2.3435  1    0.12581
nodes  -2909.0 61.6050  1  4.198e-15 ***
surg   -2906.2  5.7134  1    0.01684 *
adhere -2903.5  5.3740  1    0.02044 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Cox Markov Model: transition 0 -> 2


        loglik   Chisq Df Pr(>|Chi|)
NULL   -231.79
rx     -231.54  0.4938  2     0.7812
sex    -231.04  1.0065  1     0.3158
age    -219.26 23.5445  1  1.221e-06 ***
```

```
nodes  -218.04  2.4536  1     0.1173
surg   -217.35  1.3830  1     0.2396
adhere -217.34  0.0145  1     0.9043
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Cox Markov Model: transition 1 -> 2


        loglik   Chisq Df Pr(>|Chi|)
NULL   -1897.5
rx     -1895.0  4.8864  2  0.0868804 .
sex    -1892.8  4.3995  1  0.0359501 *
age    -1890.8  4.0650  1  0.0437799 *
nodes  -1883.4 14.7205  1  0.0001247 ***
surg   -1883.4  0.0090  1  0.9242629
adhere -1882.9  1.0505  1  0.3054007
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The effect of the continuous covariates on the log hazards is often assumed to have a linear functional form in all intensities. To introduce flexibility into the Cox Markov model, several smoothing methods may be applied, but P-splines (Eilers and Marx, 1996) are being most frequently considered in this context. Results showed a strong nonlinear effect for nodes (checked through a formal test) when using a Cox model on the recurrence transition. Figure 2 returns a centered set of predictions on a log hazard scale. The average predicted value is zero with a mean value of nodes as the reference (see the vignette 'Splines, plots, and interactions' in (Therneau, 2021)). The main curve depicts the smooth curve for nodes on a log hazard scale, indicating that the risk of recurrence increases rapidly until about 6 nodes. The apparent decrease after 23 nodes is not significant due to the wide confidence intervals.

```
> library(ggplot2)
> library(plotly)

> fit2.cmm <- coxidm(survIDM(time1, event1, Stime, event) ~ rx + sex + age +
                  pspline(nodes) + surg + adhere, data = colonIDM)


> d<-data.frame(x=fit2.cmm$term01$nodes$x, y=fit2.cmm$term01$nodes$y,
                y1=fit2.cmm$term01$nodes$y-1.96*fit2.cmm$term01$nodes$se,
                y2=fit2.cmm$term01$nodes$y+1.96*fit2.cmm$term01$nodes$se)

> nonlinear<-ggplot(d, aes(x,y))+theme(axis.text=element_text(size=13))+
             theme_bw()+labs(x = "nodes") +
             labs(y = "Partial for pspline(nodes)")+
             geom_ribbon(aes(ymin=y1,ymax=y2),fill='gray92',alpha=0.9)+
             geom_line(aes(x,y))+
             geom_line(color=1,size=1)

> ggplotly(nonlinear)
```

The proportional hazards assumption can be tested formally using the summary function. The output can be obtained putting type='ph' in summary function.

```
> summary(fit2.cmm, type = 'ph')

Cox Markov Model: transition 0 -> 1
Test the Proportional Hazards Assumption

                  chisq   df    p
rx             4.12e-01 2.00 0.81
sex            2.10e+00 1.00 0.15
age            9.37e-04 1.00 0.98
```

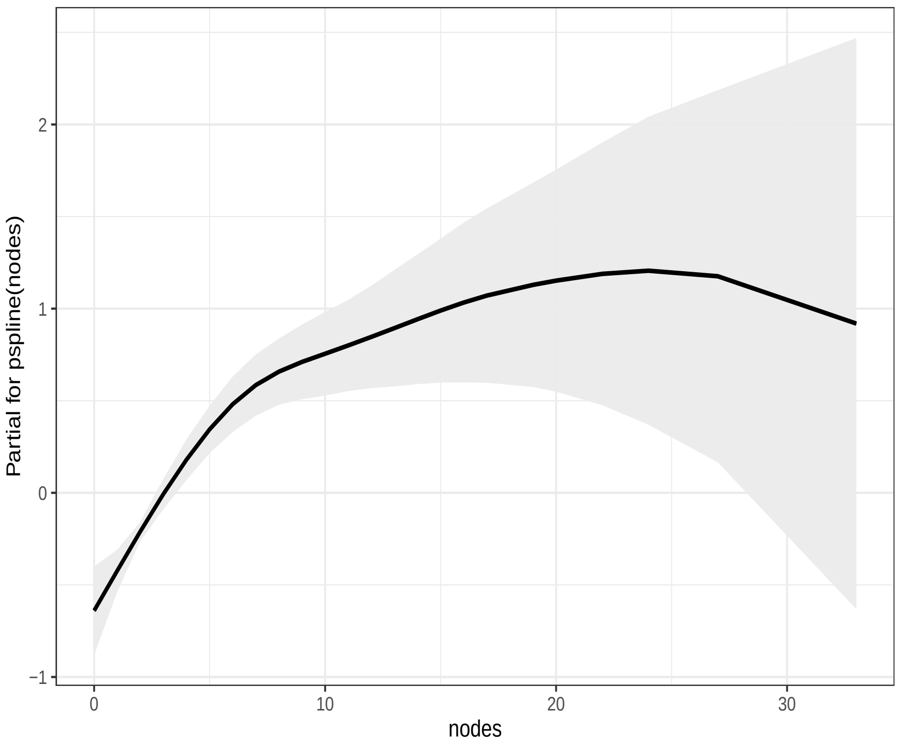


Figure 2: Predicted values of the smooth log hazard based on penalized splines (black line) with pointwise 95% confidence intervals obtained from the partial residuals for nodes (recurrence intensity), using the colon cancer data.

```
pspline(nodes) 7.60e+00 3.95 0.10
surg           1.97e+00 1.00 0.16
adhere         6.13e-01 1.00 0.43
GLOBAL         1.30e+01 9.94 0.22


Cox Markov Model: transition 0 -> 2
Test the Proportional Hazards Assumption

                chisq   df    p
rx             1.6292 2.00 0.44
sex            0.0668 1.00 0.80
age            0.8396 1.00 0.36
pspline(nodes) 0.7859 4.00 0.94
surg           0.4955 1.00 0.48
adhere         2.3606 1.00 0.12
GLOBAL         6.1424 9.99 0.80


Cox Markov Model: transition 1 -> 2
Test the Proportional Hazards Assumption

                  chisq    df    p
rx              5.03913  1.99 0.08
sex             0.02204  1.00 0.88
age             0.73628  1.00 0.39
pspline(nodes)  4.25500  4.09 0.39
surg            2.02427  1.00 0.15
adhere          0.00177  1.00 0.97
GLOBAL         13.19170 10.08 0.22
```

A semi-Markov model could be obtained by including the argument `semiMarkov = TRUE` in the coxidm function.

Occupation probabilities and transition probabilities

The occupation probabilities and the transition probabilities are key quantities of interest in multi-state models. They offer interpretable results in a simple and summarized manner.

Estimates and plots of the transition probabilities for all methods introduced in Section 2.2 can be obtained using the tprob function. The default method is the Aalen-Johansen estimator (AJ) which assumes the process to be Markovian. The presmoothed version of the Aalen-Johansen estimator (PAJ) also assumes the process to be Markovian while the remaining methods (LIDA, LM, PLM, LMAJ, and PLMAJ) are free of the Markov condition.

When one is confident of the Markov assumption, the Aalen-Johansen is preferred over the non-Markovian estimators since it reports a smaller variance in estimation. Estimates and plot for the Aalen-Johansen method can be obtained through the following input commands:

```
> tpAJ <- tprob(survIDM(time1, event1, Stime, event) ~ 1, s = 365,
                method = "AJ", conf = TRUE, data = colonIDM)

> summary(tpAJ, times=365*2:6)

Estimation of pij(s=365,t)

    t        00        01        02        11        12
  730 0.7966309 0.1300071 0.0733620 0.4686360 0.5313640
 1095 0.7192603 0.1224599 0.1582799 0.2533822 0.7466178
 1460 0.6805333 0.0884287 0.2310380 0.1335300 0.8664700
 1825 0.6444157 0.0859123 0.2696720 0.0932851 0.9067149
 2190 0.6131533 0.0774912 0.3093556 0.0632835 0.9367165

2.5%

    t        00        01        02        11        12
  730 0.7673408 0.1093487 0.0589350 0.4105298 0.4728114
 1095 0.6867036 0.1026150 0.1354061 0.2105314 0.7011204
 1460 0.6468259 0.0714743 0.2030840 0.1047501 0.8346547
 1825 0.6098804 0.0688614 0.2396632 0.0708282 0.8813846
 2190 0.5780541 0.0612090 0.2777007 0.0464018 0.9172849

97.5%

    t        00        01        02        11        12
  730 0.8270390 0.1545683 0.0913208 0.5349666 0.5971676
 1095 0.7533604 0.1461425 0.1850177 0.3049547 0.7950677
 1460 0.7159973 0.1094050 0.2628397 0.1702170 0.8994981
 1825 0.6809066 0.1071852 0.3034384 0.1228620 0.9327733
 2190 0.6503836 0.0981045 0.3446188 0.0863070 0.9565597

> autoplot(tpAJ)
```

Besides being consistent regardless the Markov condition, the landmark non-Markov estimators (LM, PLM, LMAJ, and PLMAJ) can be preferable in many situations due to their greater accuracy (smaller bias). When comparing the original nonparametric landmark estimator (LM) and the Aalen-Johansen estimator, some discrepancies are observed for $t = 730$ and $t = 1095$ (2 and 3 years, respectively). In addition to the aforementioned discrepancy between the two estimates, the plots for the two methods (Figure 3) also show that the confidence bands are narrower in the case of the Aalen-Johansen, revealing less variability for this method.

```
> tpLM <- tprob(survIDM(time1, event1, Stime, event) ~ 1, s = 365,
               method = "LM", conf = TRUE, data = colonIDM)

> summary(tpLM, times=365*2:6)

Estimation of pij(s=365,t)
```

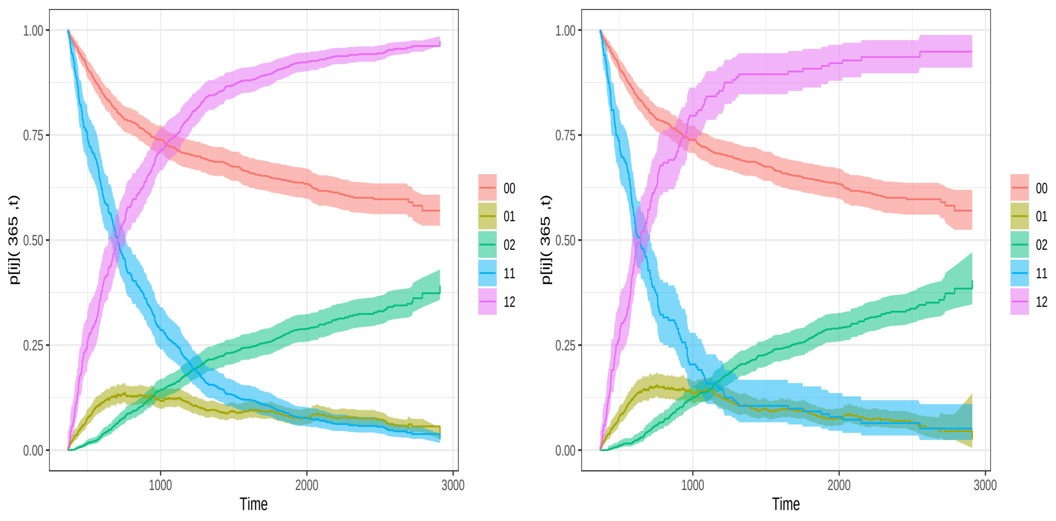


Figure 3: Transition probability estimates using the `AJ` (left hand side) and `LM` (right hand side) method, using the colon cancer data.

```
    t         00         01        02         11        12
  730 0.7966309 0.14750103 0.0558681 0.38815789 0.6118421
 1095 0.7192603 0.14320925 0.1375305 0.15789474 0.8421053
 1460 0.6805333 0.09446864 0.2249981 0.10526316 0.8947368
 1825 0.6444157 0.08583643 0.2697479 0.09210526 0.9078947
 2190 0.6131533 0.07465238 0.3121944 0.06432749 0.9356725

2.5%

    t         00         01        02         11        12
  730 0.7673274 0.12294665 0.0411836 0.31792669 0.5390734
 1095 0.6866872 0.12033558 0.1142137 0.10937624 0.7860868
 1460 0.6468058 0.07447488 0.1960521 0.06621973 0.8472552
 1825 0.6098421 0.06804756 0.2387239 0.05591405 0.8630680
 2190 0.5777125 0.05742370 0.2791810 0.03480413 0.8969820

97.5%

    t         00         01         02        11        12
  730 0.8270534 0.17695930 0.07578852 0.4739034 0.6944337
 1095 0.7533784 0.17043081 0.16560740 0.2279357 0.9021157
 1460 0.7160195 0.11982998 0.25821767 0.1673268 0.9448794
 1825 0.6809493 0.10827565 0.30480372 0.1517218 0.9550498
 2190 0.6507682 0.09705015 0.34911161 0.1188947 0.9760319

> autoplot(tpLM)
```

Since the landmark estimators of the transition probabilities are free of the Markov assumption, they can also be used to introduce such tests (at least in the scope of the illness-death model) by measuring their discrepancy to Markovian estimators. The function `markov.test` performs a local graphical test for the Markov condition. This graphical test is based on a PP-plot which compares the estimations reported by the Aalen-Johansen transition probabilities to their non-Markov counterparts. The corresponding plot for a local test of Markovianity ($s = 365$) can be obtained through the following input command:

```
> mk <- markov.test(survIDM(time1, event1, Stime, event) ~ 1, s = 365, data = colonIDM)
> autoplot(mk)
```

The plot shown in Figure 4 compares the Aalen-Johansen estimator and the landmark non-Markovian estimator for $p_{01}(s,t)$, $p_{02}(s,t)$, and $p_{12}(s,t)$, for $s = 365$. Existing deviations of the plots with respect to the straight line $y = x$ reveals some evidence on the lack of Markovianity of the underlying process beyond one year after surgery. For further illustration, this figure jointly displays

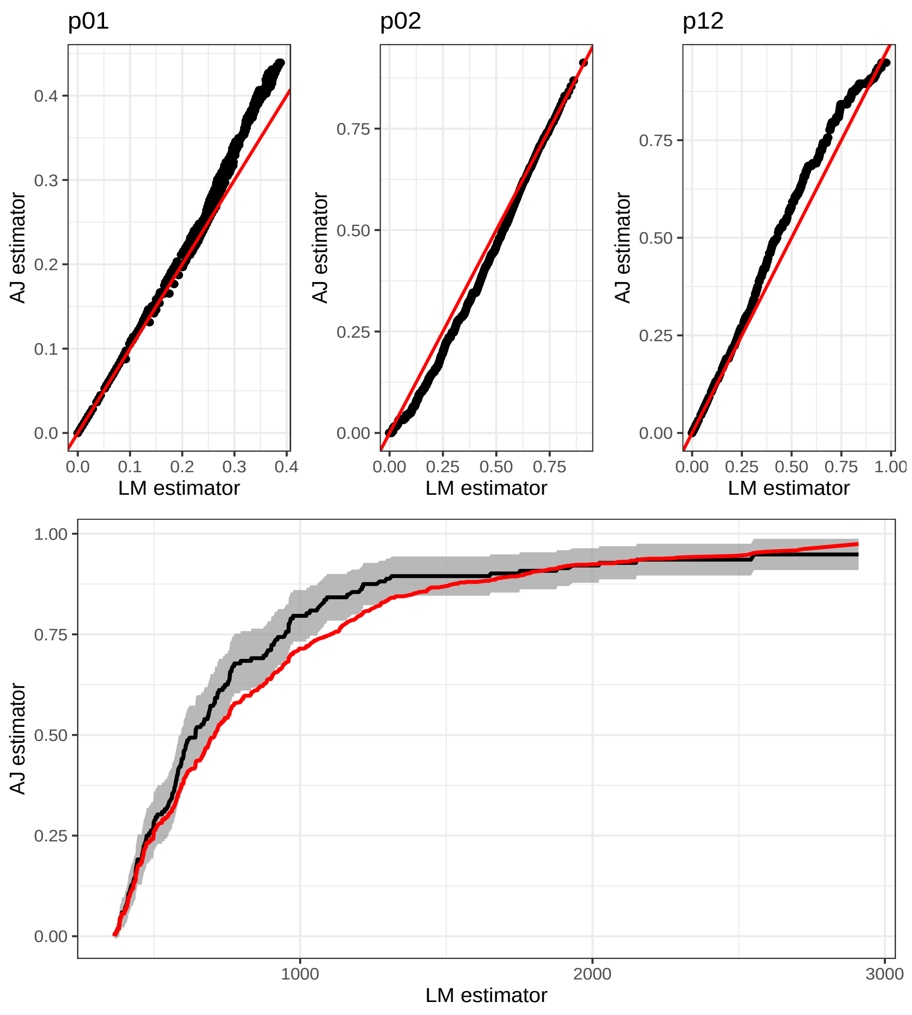


Figure 4: Graphical test for the Markov condition, $s = 365$. The second row shows the landmark (Markov-free) estimator with 95% pointwise confidence limits (black line) and Aalen-Johansen estimator (red line) for the transition probability $p_{12}(365, t)$, using the colon cancer data.

the landmark non-Markovian estimator and the Aalen-Johansen estimator for $p_{12}(s = 365, t)$. In this plot, the differences between both estimators are clearly seen. Thus, in principle, the application of the Aalen-Johansen method is not recommended here due to possible biases.

The variability of the nonparametric landmark estimator (`LM`) may be successfully reduced using presmoothing ideas (Dikta, 1998; Cao et al., 2005). The presmoothed landmark estimator is implemented in the same function through the method `PLM`. The same ideas can be used to reduce the variability of the Markovian Aalen-Johansen estimator and the (non-Markov) Landmark Aalen-Johansen estimator through methods `PAJ` and `PLMAJ`, respectively.

The package **survidm** also allows for the computation of the above quantities conditional on covariates that are observed for an individual before the individual makes a particular transition of interest. For continuous covariates, one possible and flexible nonparametric approach is to consider local smoothing by means of kernel weights based on local constant (Nadaraya-Watson: `NW`) regression. This estimator is implemented in our package through function `tprob` using the `method = IPCW`. Below are the input commands to obtain the estimates of the transition probabilities at time $s = 365$ for an individual of 48 years old. For the bandwidth in the estimator, we use `dpik` function, which is available from the R **KernSmooth** package. This is the data-based bandwidth selector of Wand and Jones (1997).

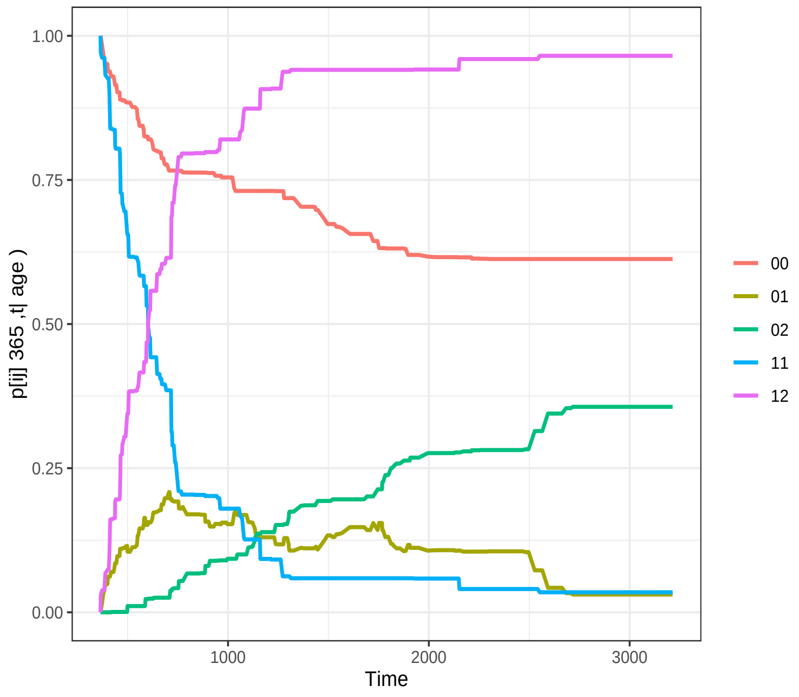


Figure 5: Conditional transition probabilities given that the subject is alive and disease-free at $s = 365$ days for a 48-years-old patient, using the colon cancer data.

```
> tpIPCW.age <- tprob(survIDM(time1, event1, Stime, event) ~ age, s = 365,
                method = "IPCW", z.value = 48, conf = FALSE, data = colonIDM,
                bw = "dpik", window = "gaussian", method.weights = "NW")

> summary(tpIPCW.age, time=365*2:6)

Estimation of pij(s=365,t)

    t        00        01         02         11        12
  730 0.7662208 0.1921290 0.04165012 0.28946129 0.7105387
 1095 0.7308496 0.1688189 0.10033149 0.12631010 0.8736899
 1460 0.6980293 0.1088373 0.19313342 0.05905711 0.9409429
 1825 0.6310625 0.1186104 0.25032706 0.05903929 0.9409607
 2190 0.6157095 0.1051797 0.27911080 0.04035816 0.9596418

> autoplot(tpIPCW.age)
```

The curves depicted in Figure 5, which are purely nonparametric, enable flexible modeling of the data providing flexible and robust effects of the covariate that can be used at least as a preliminary attempt, providing insights on the data being analyzed. Such methods can be used to capture nonstandard data features that may not be detected through parametric or semiparametric proposals. A general problem in multivariate nonparametric regression estimation is the so-called curse of dimensionality. In higher dimensions, the observations are sparsely distributed even for large sample sizes. Consequently, estimators based on local averaging (like those based on kernel smoothing) perform unsatisfactorily in this situation.

An alternative method is to consider estimators based on Cox's regression model (Cox, 1972) fitted marginally to each transition with the corresponding baseline hazard function estimated by Breslow's method (Breslow, 1972). The following input commands illustrate the use of the `tprob` function in this context:

```
> tp.breslow.age <- tprob(survIDM(time1, event1, Stime, event) ~ age, s = 365,
                    method = "breslow", z.value = 48, conf = FALSE, data = colonIDM)

> summary(tp.breslow.age, time=365*2:6)
```

```
Estimation of pij(s=365,t)

    t        00         01         02         11        12
  730 0.7970855 0.15020199 0.05271253 0.37528949 0.6247105
 1095 0.7198657 0.14999685 0.13013746 0.14814634 0.8518537
 1460 0.6826444 0.10384005 0.21351550 0.09843946 0.9015605
 1825 0.6451532 0.09850122 0.25634562 0.08617378 0.9138262
 2190 0.6139465 0.08891388 0.29713961 0.06066618 0.9393338
```

Note that if the argument `z.value` is missing, then the `tprob` function computes the predicted conditional transition probabilities at the average values of the covariate. The Breslow method (based on the Cox regression model) is particularly well-suited to the setting with multiple covariates:

```
> tp.breslow <- tprob(survIDM(time1, event1, Stime, event) ~ rx + age + nodes, s = 365,
                method = "breslow", z.value = c('Obs', 50, 10), conf = FALSE,
                data = colonIDM)

> summary(tp.breslow, time=365*2:6)

Estimation of pij(s=365,t)

    t        00         01        02         11        12
  730 0.6423398 0.24905912 0.1086010 0.30017412 0.6998259
 1095 0.5222992 0.21890332 0.2587975 0.09465150 0.9053485
 1460 0.4680828 0.12787851 0.4040387 0.05433167 0.9456683
 1825 0.4181094 0.10712224 0.4747684 0.04519157 0.9548084
 2190 0.3762996 0.08424903 0.5394514 0.02685212 0.9731479
```

Cumulative Incidence Function

Another quantity of interest in multi-state modeling is the cause-specific cumulative incidence of the illness (recurrence). Function `CIF` can be used to obtain the nonparametric estimator of Geskus (2011) (default method), which is equivalent to the classical Aalen-Johansen estimator. The corresponding presmoothed version (Meira-Machado and Sestelo, 2018) is also implemented through the argument `presmooth = TRUE`:

```
> cif <- CIF(survIDM(time1, event1, Stime, event) ~ 1, data = colonIDM, conf = TRUE)
> summary(cif, time=365*1:6)

Estimation of CIF(t)
    t       CIF
  365 0.2378902
  730 0.3844412
 1095 0.4372663
 1460 0.4620841
 1825 0.4859813
 2190 0.5032043

2.5%

    t       CIF
  365 0.2088267
  730 0.3509039
 1095 0.4038141
 1460 0.4296740
 1825 0.4540347
 2190 0.4697608

97.5%

    t       CIF
  365 0.2616792
  730 0.4103338
 1095 0.4666664
```

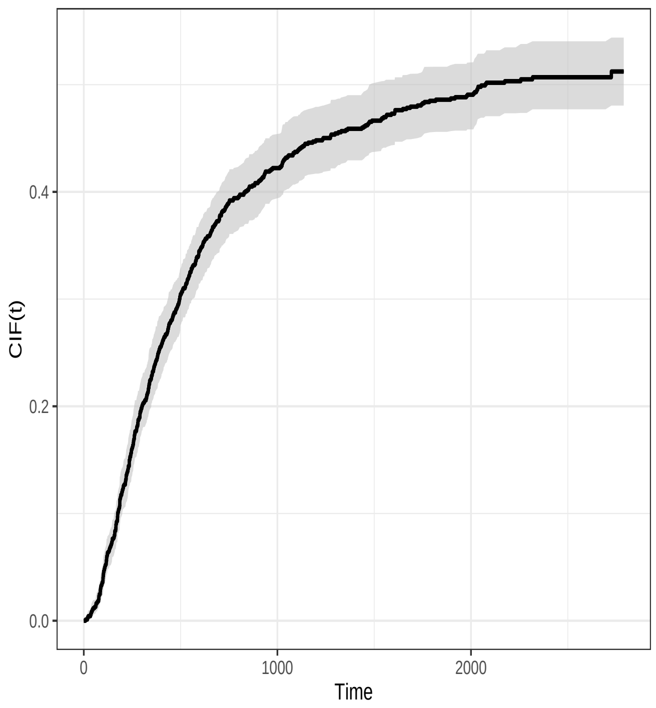


Figure 6: Cumulative incidence function in the recurrence state with 95% bootstrap confidence bands, using the colon cancer data.

```
 1460 0.4900876
 1825 0.5161684
 2190 0.5319749

> autoplot(cif, ylim=c(0, 0.6), confcol = 2)
```

Figure 6 depicts the estimates of cumulative incidence function for the recurrent state together with a 95% pointwise confidence bands based on simple bootstrap that resamples each datum with probability $1/n$. From this plot, it can be seen that individuals have a probability of recurrence higher than 50%. This cumulative probability is about 43% at three years after surgery.

Figure 7 depicts the estimates of the (conditional) cumulative incidence function for patients with 1 and 9 lymph nodes with detectable cancer. Curves depicted in this figure, which are purely nonparametric, indicate that patients with 9 lymph nodes with detectable cancer have a considerably higher probability of recurrence. The corresponding input commands are shown below:

```
> cif.1.nodes <- CIF(survIDM(time1, event1, Stime, event) ~ nodes, data = colonIDM,
         conf = FALSE, z.value = 1)
> cif.9.nodes <- CIF(survIDM(time1, event1, Stime, event) ~ nodes, data = colonIDM,
                 conf = FALSE, z.value = 9)

> d<-as.data.frame(cbind(rep(cif.1.nodes$est[,1],2),c(cif.1.nodes$est[,2],
                cif.9.nodes$est[,2]), c(rep("1 nodes", length(cif.1.nodes$est[,1])),
                rep("9 nodes", length(cif.1.nodes$est[,2])))))

> names(d)<-c('time','cif','type')

> cif<-ggplot(d, aes(x=as.numeric(time), y=as.numeric(cif),group=factor(type),
                 color=factor(type)))+theme_bw()+labs(x = 'Time (days)',
                 y = 'CIF(t|nodes)')
> cif+geom_step(size=1)+ theme(legend.title=element_blank())
```

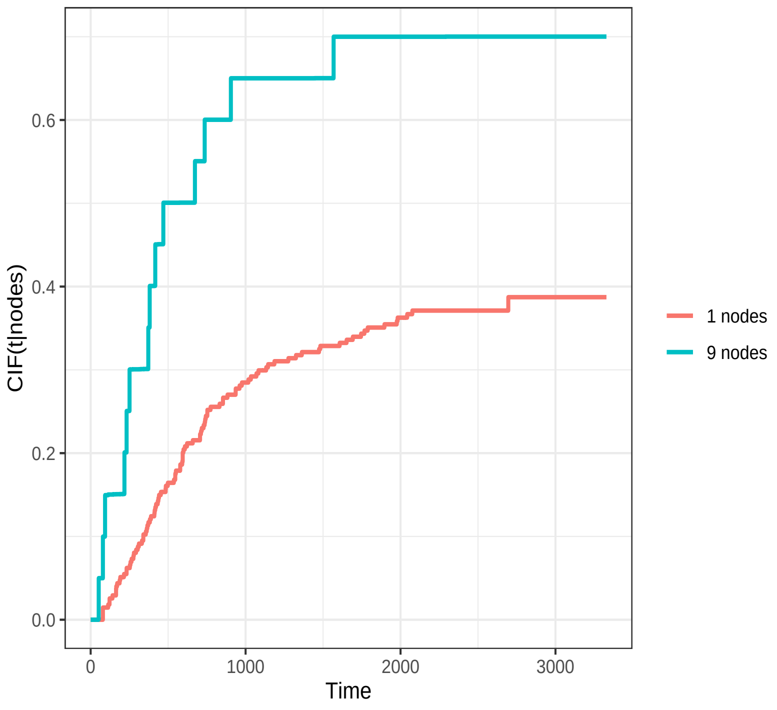


Figure 7: Conditional cumulative incidence function for the colon cancer data for `nodes = 1` and `nodes = 9`, using the colon cancer data.

Sojourn distribution

Another interesting quantity is the sojourn time in each state. Estimates for the distribution function of the sojourn time in the recurrence state can be obtained using the estimator by Satten and Datta (2002) through function `sojourn`.

```
> soj <- sojourn(survIDM(time1, event1, Stime, event) ~ 1,
               data = colonIDM, method = "Satten-Datta", conf = FALSE)
> summary(soj, time=365*1:6)

Estimation of sojourn(t)

    t   sojourn
  365 0.4852424
  730 0.7723636
 1095 0.8755021
 1460 0.8983714
 1825 0.9102335
 2190 0.9220849
```

The estimates for the distribution function of the sojourn time in the recurrence state, corresponding to the time between entry in recurrence and death, reveal that the distribution function increases to a value near 49% and 78% for a time of one and two years, respectively, revealing a high risk of death shortly after relapse.

The methods for implementing some of the proposed methods can be computationally demanding. In particular, the use of bootstrap resampling techniques is time-consuming process because it is necessary to estimate the model a great number of times. In such cases, we recommend the use of parallelization (`cluster = TRUE`). This should considerably increase performance on multi-core/multi-threading machines.

Conclusions

There has been several recent contributions for the inference in the context of multi-state models. Many of these contributions were made for the illness-death model. One important and perhaps undervalued aspect of multi-state models is the possibility to apply them to obtain predictions of the clinical prognosis. This is usually achieved using estimates of the transition probabilities and survival estimates. However, there are several other quantities that could also be used in the analysis of these data, such as the state occupation probabilities, the sojourn time distributions, and the cumulative incidence functions. To provide the biomedical researchers with an easy-to-use tool for obtaining predictive estimates for all these quantities, we develop an R package called **survidm**. This package can be used to implement several nonparametric and semiparametric estimators for the transition probabilities. In addition, estimators have also implemented that account for the influence of covariates. Bootstrap confidence bands are provided for all methods. The software can also be used to perform multi-state regression (using type-specific Cox models).

One limitation of the **survidm** R package is that it can only be used in the progressive illness-death model. However, this turns out to be an advantage for those users that only wish to analyze data from a progressive illness-death model. For such cases, the **survidm** package is ideal since it is user-friendly (as illustrated in the real data analysis as well as in the help files of the main functions `tprob`, `sojourn`, and `CIF`) with a strong resemblance to the well-known and widely used **survival** package.

Acknowledgments

This research was financed by Portuguese Funds through FCT - "Fundação para a Ciência e a Tecnologia", within the research grant PD/BD/142887/2018. Luís Meira-Machado acknowledges financial support from the Spanish Ministry of Economy and Competitiveness MINECO through project MTM2017-82379-R funded by (AEI/FEDER, UE) and acronym "AFTERAM".

Bibliography

O. Aalen and S. Johansen. An empirical transition matrix for non homogeneous markov and chains based on censored observations. *Scandinavian Journal of Statistics*, 5:141–150, 1978. [p72]

O. O. Aalen, O. Borgan, and H. Fekjaer. Covariate adjustment of event histories estimated from markov chains: The additive approach. *Biometrics*, 57(4):993–1001, 2001. ISSN 0006-341X. [p72]

A. Allignol and A. Latouche. Cran task view: Survival analysis. *Version 2018-05-04, URL http://CRAN.R-project.org/view=Survival*, 2018. [p71]

A. Allignol, M. Schumacher, and J. Beyersmann. Empirical transition matrix of multi-state models: The etm package. *Journal of Statistical Software*, 38:4:1–15, 2011. [p71]

P. K. Andersen, Ø. Borgan, R. D. Gill, and N. Keiding. *Statistical Models Based on Counting Processes*. Springer-Verlag, New York, 1993. [p70]

A. A. Araújo, J. Roca-Pardiñas, and L. Meira-Machado. Tpmsm: Estimation of the transition probabilities in 3-state models. *Journal of Statistical Software*, 62:1–29, 2014. [p71]

V. Balboa and J. de Uña-Álvarez. Estimation of transition probabilities for the illness-death model: Package tp.idm. *Journal of Statistical Software*, 83:10:1–19, 2018. [p71]

Ø. Borgan. Aalen-johansen estimator. *Encyclopedia of Biostatistics*, 1:5–10, 1988. [p72]

N. Breslow. Discussion of paper by dr cox. *Journal of Royal Statistical Society, Series B*, pages 216–217, 1972. [p73, 82]

R. Cao, I. Lopez-de Ullibarri, P. Janssen, and N. Veraverbeke. Presmoothed kaplan-meier and nelson-aalen estimators. *Journal of Nonparametric Statistics*, 17:31–56, 2005. [p72, 81]

D. Cox. Regression models and life tables. *Journal of the Royal Statistical Society Series B*, 34:187–220, 1972. [p73, 82]

S. Datta and G. Satten. Validity of the aalen-johansen estimators of stage occupation probabilities and nelson aalen integrated transition hazards for non-markov models. *Statistics & Probability Letters*, 55: 403–411, 2001. [p72, 73]

J. de Uña-Álvarez and L. Meira-Machado. Nonparametric estimation of transition probabilities in the non-markov illness-death model: A comparative study. *Biometrics*, 71(2):364–375, 2015. ISSN 0006-341X. [p71, 72]

L. de Wreede, M. Fiocco, and H. Putter. mstate: An r package for the analysis of competing risks and multi-state models. *Journal of Statistical Software*, 38:7:1–30, 2011. [p71]

G. Dikta. On semiparametric random censorship models. *Journal of Statistical Planning and Inference*, 66:253–279, 1998. [p72, 81]

P. Eilers and B. Marx. Flexible smoothing with b-splines and penalties. *Statistical Science*, 11:89–121, 1996. [p77]

R. B. Geskus. Cause-specific cumulative incidence estimation and the fine and gray model under both left truncation and right censoring. *Biometrics*, 67(1):39–49, 2011. ISSN 0006-341X. [p73, 83]

P. Hougaard. *Analysis of Multivariate Survival Data*. Statistics for Biology and Health. Springer-Verlag, New York, 2000. [p70]

C. Jackson. Multi-state models for panel data: The msm package for r. *Journal of Statistical Software*, 38:8:1–28, 2011. [p72]

J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*. John Wiley & Sons, 1980. [p73]

E. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958. [p72]

L. Meira-Machado. Smoothed landmark estimators of the transition probabilities. *SORT-Statistics and Operations Research Transactions*, 40(2):375–398, Jul-Dec 2016. ISSN 1696-2281. [p72]

L. Meira-Machado and J. Roca-Pardiñas. p3state.msm: Analyzing survival data from an illness-death model. *Journal of Statistical Software*, 38:3, 2011. [p71]

L. Meira-Machado and M. Sestelo. Estimation in the progressive illness-death model: A nonexhaustive review. *Biometrical Journal*, 61:245–263, 2019. doi: 10.1002/bimj.201500038. [p70, 71, 73]

L. Meira-Machado, J. de Uña-Álvarez, and C. Cadarso-Suárez. Nonparametric estimation of transition probabilities in a non-markov illness-death model. *Lifetime Data Analysis*, 12:325–344, 2006. [p72]

L. Meira-Machado, J. de Uña-Álvarez, C. Cadarso-Suárez, and P. Andersen. Multi-state models for the analysis of time to event data. *Statistical Methods in Medical Research*, 18:195–222, 2009. [p70]

L. Meira-Machado, J. de Uña-Álvarez, and S. Datta. Nonparametric estimation of conditional transition probabilities in a non-markov illness-death model. *Computational Statistics*, 30:377–397, 2015. [p73]

C. G. Moertel, T. R. Fleming, J. S. Macdonald, D. G. Haller, J. A. Laurie, P. J. Goodman, J. S. Ungerleider, W. A. Emerson, D. C. Tormey, J. H. Glick, M. H. Veeder, and J. A. Mailliard. Levamisole and fluorouracil for adjuvant therapy of resected colon carcinoma. *New England Journal of Medicine*, 322 (6):352–358, 1990. [p74]

A. Moreira, J. de Uña-Álvarez, and L. Meira-Machado. Presmoothing the aalen-johansen estimator in the illness-death model. *Electronical Journal of Statistics*, 7:1491–1516, 2013. ISSN 1935-7524. [p72]

H. Putter and C. Spitoni. Non-parametric estimation of transition probabilities in non-markov multi-state models: The landmark aalen-johansen estimator. *Statistical Methods in Medical Research*, 27: 2081–2092, 2018. [p73]

H. Putter, M. Fiocco, and R. B. Geskus. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine*, 26(11):2389–2430, mayo 2007. ISSN 0277-6715. [p70]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL `https://www.r-project.org/`. [p73]

M. Rodríguez-Álvarez, L. Meira-Machado, and E. Abu-Assi. Nonparametric estimation of time-dependent roc curves conditional on a continuous covariate. *Statistics in medicine*, 35:7:1090–1102, 2016. [p73]

G. A. Satten and S. Datta. Marginal estimation for multi-stage models: waiting time distributions and competing risks analyses. *Statistics in Medicine*, 21(1):3–19, 2002. ISSN 0277-6715. [p73, 85]

T. M. Therneau. *A Package for Survival Analysis in R*, 2021. URL `https://CRAN.R-project.org/package=survival`. R package version 3.2-11. [p77]

H. C. Van Houwelingen. Dynamic prediction by landmarking in event history analysis. *Scandinavian Journal of Statistics*, 34(1):70–85, marzo 2007. ISSN 0303-6898. [p72]

M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hal, London, 1997. [p81]

L. Wei. The accelerated failure time model: a useful alternative to the cox regression model in survival analysis. *Statist. Med.*, 11:1871–1879, 1992. [p72]

Gustavo Soutinho
Institute of Public Health of the University of Porto (ISPUP)
University of Porto
Rua das Taipas no. 135, 4050-600, Porto, Portugal
ORCID: 0000-0002-0559-1327
gdsoutinho@gmail.com

Marta Sestelo
Department of Statistics and OR,
SiDOR research group & CITMaga,
University of Vigo
Campus Lagoas-Marcosende, 36310 - Vigo, Spain
ORCID: 0000-0003-4284-6509
sestelo@uvigo.es

Luís Meira-Machado
Department of Mathematics & Centre of Mathematics
University of Minho
Campus de Azurém, 4800-058, Guimarães, Portugal
ORCID: 0000-0002-8577-7665
lmachado@math.uminho.pt

Multiple Imputation and Synthetic Data Generation with NPBayesImputeCat

by Jingchen Hu, Olanrewaju Akande and Quanli Wang

Abstract In many contexts, missing data and disclosure control are ubiquitous and challenging issues. In particular, at statistical agencies, the respondent-level data they collect from surveys and censuses can suffer from high rates of missingness. Furthermore, agencies are obliged to protect respondents' privacy when publishing the collected data for public use. The **NPBayesImputeCat** R package, introduced in this paper, provides routines to i) create multiple imputations for missing data and ii) create synthetic data for statistical disclosure control, for multivariate categorical data, with or without structural zeros. We describe the Dirichlet process mixture of products of the multinomial distributions model used in the package and illustrate various uses of the package using data samples from the American Community Survey (ACS). We also compare results of the missing data imputation to the **mice** R package and those of the synthetic data generation to the **synthpop** R package.

Introduction and background

Multiple imputation for missing data

Missing data problems arise in many statistical analyses. To impute missing values, multiple imputation, first proposed by Rubin (1987), has been widely adopted. This approach estimates predictive models based on the observed data, fills in missing values with draws from the predictive models, and produces multiple imputed and completed datasets. Data analysts then apply standard statistical analyses (e.g., regression analysis) on each imputed dataset and use appropriate combining rules to obtain valid point estimates and variance estimates (Rubin, 1987).

As a brief review of the multiple imputation combining rules for missing data, let q be the completed data estimator of some estimand of interest Q, and let u be the estimator of the variance of q. For $l = 1, \ldots, m$, let $q^{(l)}$ and $u^{(l)}$ be the values of q and u in the lth completed dataset. The multiple imputation estimate of Q is equal to $\bar{q}_m = \sum_{l=1}^{m} q^{(l)}/m$, and the estimated variance associated with $\bar{q}_m$ is equal to $T_m = (1 + 1/m)b_m + \bar{u}_m$, where $b_m = \sum_{l=1}^{m}(q^{(l)} - \bar{q}_m)^2/(m-1)$ and $\bar{u}_m = \sum_{l=1}^{m} u^{(l)}/m$. Inferences for Q are based on $(\bar{q}_m - Q) \sim t_v(0, T_m)$, where t_v is a t-distribution with $v = (m-1)(1 + \bar{u}_m/[(1+1/m)b_m])^2$ degrees of freedom.

Multiple imputation by chained equations (MICE, Buuren and Groothuis-Oudshoorn (2011)) remains the most popular method for generating multiple completed datasets after multiple imputation. Under MICE, one specifies univariate conditional models separately for each variable, usually using generalized linear models (GLMs) or classification and regression trees (CART Breiman et al. (1984); Burgette and Reiter (2010)), and then iteratively samples plausible predicted values from the sequence of conditional models . For implementing MICE in R, most analysts use the **mice** package. For an in-depth review of the MICE algorithm, see Buuren and Groothuis-Oudshoorn (2011). For more details and reviews, see Rubin (1996), Harel and Zhou (2007), Reiter and Raghunathan (2007).

Synthetic data for statistical disclosure control

Statistical agencies regularly collect information from surveys and censuses and make such information publicly available for various purposes, including research and policymaking. In numerous countries around the world, statistical agencies are legally obliged to protect respondents' privacy when making this information available to the public. Statistical disclosure control (SDC) is the collection of techniques applied to confidential data before public release for privacy protection. Popular SDC techniques for tabular data include cell suppression and adding noise, and popular SDC techniques for respondent-level data (also known as microdata) include swapping, adding noise, and aggregation. Hundepool et al. (2012) provide a comprehensive review of SDC techniques and applications.

The multiple imputation methodology has been generalized to SDC. One approach to facilitating microdata release is to provide synthetic data. First proposed by Little (1993) and Rubin (1993), the synthetic data approach estimates predictive models based on the original, confidential data, simulates synthetic values with draws from the predictive models, and produces multiple synthetic datasets. Data analysts then apply standard statistical analyses (e.g., regression analysis) on each synthetic dataset and use appropriate combining rules (different from those in multiple imputation) to obtain valid point estimates and variance estimates (Reiter and Raghunathan, 2007; Drechsler, 2011). Moreover, synthetic data comes in two flavors: fully synthetic data (Rubin, 1993), where every variable is deemed sensitive and therefore synthesized, and partially synthetic data (Little, 1993), where only a subset of variables is deemed sensitive and synthesized, while the remaining variables are un-synthesized. Statistical agencies can choose between these two approaches depending on their protection goals, and subsequent analyses also differ.

When dealing with fully synthetic data, $\bar{q}_m$ estimates Q as in the multiple imputation setting, but the estimated variance associated with $\bar{q}_m$ becomes $T_f = (1 + 1/m)b_m - \bar{u}_m$, where b_m and $\bar{u}_m$ are defined as in previous section on multiple imputation. Inferences for Q are now based on $(\bar{q}_m - Q) \sim t_v(0, T_f)$, where the degrees of freedom is $v_f = (m-1)(1 - m\bar{u}_m/((m+1)b_m))^2$.

For partially synthetic data, $\bar{q}_m$ still estimates Q but the estimated variance associated with $\bar{q}_m$ is $T_p = b_m/m + \bar{u}_m$, where b_m and $\bar{u}_m$ are defined as in the multiple imputation setting. Inferences for Q are based on $(\bar{q}_m - Q) \sim t_v(0, T_p)$, where the degrees of freedom is $v_p = (m-1)(1 + \bar{u}_m/[b_m/m])^2$.

For synthetic data with R, **synthpop** provides synthetic data generated by drawing from conditional distributions fitted to the confidential data. The conditional distributions are estimated by models chosen by the user, whose choices include parametric or CART models. For more details and reviews of synthetic data for statistical disclosure control, see Drechsler (2011).

Structural zeros

An important feature of survey data is the existence of structural zeros, which are combinations of variables with probability zero. For example, in the combinations of variables of vital signs, there should not exist a deceased patient with a pulse. For the household surveys, in the combinations of variables of relationship and age, there should not exist a household where a son is older than his biological father. As another example, if a dataset contains information of a record's age and educational attainment in the form of categorical variables, there can be no record having the combination of being younger than 5 and having a doctorate degree.

In survey data with many variables, cross-tabulations of variables could result in sparse tables, containing non-structural zeros (combinations that are possible but happen not to exist in the particular dataset) and structural zeros (combinations that are simply impossible). To deal with structural zeros, many advanced statistical models are designed to assign zero probability for every impossible combination, which is a challenging task.

What NPBayesImputeCat does

The **NPBayesImputeCat** package specializes in estimating and performing multiple imputation and synthetic data generation for multivariate categorical data. Unlike **mice** and **synthpop**, both of which specify conditional models, the **NPBayesImputeCat** implements the Dirichlet process mixture of products of multinomial distributions (DPMPM), which specifies a joint latent class model on multivariate categorical variables. It uses Dirichlet process (DP) priors to allow effective clustering of the observations. Therefore, the **NPBayesImputeCat** package adds to the tools of imputation and synthesis, where a joint model might be more suitable than a series of conditional models for multivariate categorical data.

NPBayesImputeCat also allows imputation with structural zeros. It, therefore, helps fill an important gap in missing data imputation techniques, as currently available R packages do not facilitate imputation with structural zeros, and users might have to post-process, such as rejection sampling to delete generated but impossible cases.

For multiple imputation, the **NPBayesImputeCat** package allows data with and without structural zeros. For synthetic data, currently, the package only allows data without structural zeros.

The structure of this paper

The rest of the paper is organized as follows. We first introduce the joint latent class models for multivariate categorical data that the **NPBayesImputeCat** package applies, that is, the DPMPM model. In addition, we review applications of multiple imputation and synthetic data generation using the DPMPM in the literature. Next, we introduce the sample datasets from the American Community Survey (ACS) to be used in the demonstration, and provide illustrations for both multiple imputation and synthetic data generation using the **NPBayesImputeCat** package while comparing to other existing R packages. The paper concludes with a summary and discussion.

The DPMPM model

Proposed by Dunson and Xing (2009), the DPMPM is a Bayesian latent class model developed for multivariate categorical data. To allow for effective clustering of the observations based on all categorical variables, DP priors are specified for the mixture probabilities and multinomial probability vectors of the categorical data. The DPMPM has been shown to capture the complex dependencies in multivariate categorical data while being computationally efficient. In addition, it empowers the data to select the number of latent classes to be used in the model estimation. The model has also been extended to account for structural zeros in categorical data (Manrique-Vallier and Reiter, 2014a).

The **NPBayesImputeCat** package includes two versions of the DPMPM: i) DPMPM without structural zeros, and ii) DPMPM with structural zeros. In this section, we introduce the details of both versions and review previous work on using the DPMPM for multiple imputation and synthetic data.

DPMPM without structural zeros

Our review of the DPMPM without structural zeros closely follows the review in Hu and Hoshino (2018). Consider a sample $\mathbf{X}$ consisting of n records, where each ith record, with $i = 1, \ldots, n$, has p unordered categorical variables. The basic assumption of the DPMPM is that every record $\mathbf{X}_i = (X_{i1}, \cdots, X_{ip})$ belongs to one of K underlying unobserved/latent classes. Given the latent class assignment z_i of record i, as in Equation (2), each variable X_{ij} independently follows a multinomial distribution, as in Equation (1), where d_j is the number of categories of variable j, and $j = 1, \ldots, p$.

$$X_{ij} \mid z_i, \theta \overset{ind}{\sim} \text{Multinomial}(\theta^{(j)}_{z_i 1}, \ldots, \theta^{(j)}_{z_i d_j}; 1) \quad \forall i, j \tag{1}$$

$$z_i \mid \pi \sim \text{Multinomial}(\pi_1, \ldots, \pi_K; 1) \quad \forall i \tag{2}$$

The marginal probability $Pr(X_{i1} = x_{i1}, \cdots, X_{ip} = x_{ip} \mid \pi, \theta)$ can be expressed as averaging over the latent classes:

$$Pr(X_{i1} = x_{i1}, \cdots, X_{ip} = x_{ip} \mid \pi, \theta) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{p} \theta^{(j)}_{k x_{ij}}. \tag{3}$$

As pointed out in Si and Reiter (2013), Hu et al. (2014), Akande et al. (2017), such averaging over latent classes results in dependence among the variables. Equation (3) will also help illustrate the DPMPM with structural zeros in the next section.

The DPMPM clusters records with similar characteristics based on all p variables. Relationships among all the variables are induced by integrating out the latent class assignment z_i. To empower the DPMPM to pick the effective number of occupied latent classes, the truncated stick-breaking representation (Sethuraman, 1994) of the DP prior is used as in Equation (4) through Equation (7),

$$\pi_k = V_k \prod_{l<k} (1 - V_l) \quad \text{for } k = 1, \ldots, K, \tag{4}$$

$$V_k \overset{iid}{\sim} \text{Beta}(1, \alpha) \quad \text{for } k = 1, \ldots, K-1, \quad V_K = 1, \tag{5}$$

$$\alpha \sim \text{Gamma}(a_\alpha, b_\alpha), \tag{6}$$

$$\boldsymbol{\theta}^{(j)}_k = (\theta^{(j)}_{k1}, \ldots, \theta^{(j)}_{kd_j}) \sim \text{Dirichlet}(a^{(j)}_1, \ldots, a^{(j)}_{d_j}) \quad \text{for } j = 1, \ldots, p, \quad k = 1, \ldots, K. \tag{7}$$

and a blocked Gibbs sampler is implemented for the Markov chain Monte Carlo (MCMC) sampling procedure (Ishwaran and James, 2001; Si and Reiter, 2013; Hu et al., 2014; Akande et al., 2017; Manrique-Vallier and Hu, 2018; Drechsler and Hu, 2021; Hu and Savitsky, 2018).

When used as an imputation engine, missing values are handled within the Gibbs sampler. As described in Akande et al. (2017), at one MCMC iteration l, one samples a value of the latent class indicator z_i using Equation (2), given a draw of the parameters and observed data. In this iteration l, given the sampled z_i, one samples missing values using independent draws from Equation (1). This process is repeated for every missing value in the dataset in iteration l, obtaining one imputed dataset.

When used as a data synthesizer, the fully observed confidential dataset is used for model estimation through MCMC, and sensitive variable values are synthesized as an extra step at chosen MCMC iteration. For example, at MCMC iteration l, one samples a value of the latent class indicator z_i using Equation (2). Given the sampled z_i, one samples synthetic values of sensitive variables using independent draws from Equation (1). This process is repeated for every record that has sensitive values to be synthesized, obtaining one synthetic dataset.

DPMPM with structural zeros

When structural zeros are present, we need to modify the likelihood to enforce zero probability for impossible combinations. That is, we need to truncate the support of the DPMPM. Following the general description in Manrique-Vallier and Reiter (2014a) and Manrique-Vallier and Hu (2018), let $\mathcal{C}$ represent all combinations of individuals, including impossible combinations; let $\mathcal{MCZ} \not\subseteq \mathcal{C}$ be the set of impossible combinations to be excluded. We restrict $\mathbf{X}$ to the set $\mathcal{C} \setminus \mathcal{MCZ}$, with $Pr(\mathbf{X} \in \mathcal{MCZ}) = 0$. The marginal probability in the DPMPM without structural zeros in Equation (3) then becomes

$$Pr(\mathbf{X}_i = \mathbf{x}_i \mid \pi, \theta, \mathcal{MCZ}) \propto I(\mathbf{X}_i \notin \mathcal{MCZ}) \sum_{k=1}^{K} \pi_k \prod_{j=1}^{p} \theta_{kx_{ij}}^{(j)}. \tag{8}$$

Let $\mathcal{X}^*$ be the sample that only contains possible combinations, we have the joint likelihood as

$$p(\mathcal{X}^* \mid \pi, \theta, \mathcal{MCZ}) \propto \prod_{i=1}^{n} I(\mathbf{X}_i \notin \mathcal{MCZ}) \sum_{k=1}^{K} \pi_k \prod_{j=1}^{p} \theta_{kx_{ij}}^{(j)}. \tag{9}$$

To get the Gibbs sampler to work, we follow the general data augmentation technique proposed by Manrique-Vallier and Reiter (2014a) and assume the existence of an observed sample $\mathcal{X}^0$ of unknown size $Nmis$, generated from the DPMPM without structural zeros (i.e., the unrestricted DPMPM). $\mathcal{X}^0$ only contains records that fall into $\mathcal{MCZ}$.

The same set of DP priors in Equation (4) through Equation (7) is used in the DPMPM with structural zeros. In the Gibbs sampler, we keep the generated $\mathcal{X}^0$ and combine it with $\mathcal{X}^*$ when estimating the model parameters. For computational expedience, we set the upper bound of the number of observations, $Nmis$, that can be generated in $\mathcal{X}^0$, to be fixed at a large $Nmax$ at every iteration. When used as either an imputation engine or a data synthesizer, missing values or synthetic data are generated from the truncated likelihood Equation (9).

Applications of DPMPM for multiple imputation

The DPMPM has been adapted as a multiple imputation engine to deal with missing values in categorical data. Some imputation applications have focused on the DPMPM without structural zeros, while others have dealt with the DPMPM with structural zeros.

Among the work on multiple imputation using the DPMPM without structural zeros, Si and Reiter (2013) applied the DPMPM imputation model to impute missing background data (categorical) in the 2007 Trends in International Mathematics and Science Study (TIMSS). The 2007 TIMSS data contains 80 background variables on 90,505 students. Among the 80 categorical background variables, 68 have less than 10% missing values, 6 variables have between 10% and 30% missing values, and 1 variable has more than 75% missing values.

Akande et al. (2017) designed simulation studies using data from the American Community Survey (ACS) and compared the DPMPM imputation engine to two other widely used multiple imputation engines: i) chained equations using generalized linear models, and ii) chained equations using classification and regression trees (CART). From a population of 671,153 housing units and 35 categorical variables collected and cleaned from the 2012 ACS data, Akande et al. (2017) performed repeated sampling and empirically compared the three multiple imputation models.

Among the work on multiple imputation using the DPMPM with structural zeros, Manrique-Vallier and Reiter (2014b) followed the data augmentation approach Manrique-Vallier and Reiter (2014a), and imputed missing data of repeated samples from the 5% public use microdata sample from the 2000 United States Census for the state of New York, a population of 953,076 individuals and 10 categorical variables, with the number of levels ranging from 2 to 11.

Finally, Murray (2018) provides an excellent review of practical and theoretical findings of multiple imputation research and highlights the DPMPM imputation engine as a recent development.

Applications of DPMPM for synthetic data

The DPMPM has also been used as a synthetic data generator to the public release of useful and private micro-level categorical data. Some work focused on the DPMPM without structural zeros, while others dealt with synthetic data problems using the DPMPM with structural zeros.

Among the work on synthetic data generation using the DPMPM without structural zeros, Hu et al. (2014) used the DPMPM to generate fully synthetic data for a subset of 10,000 individuals and 14 categorical variables from the 2012 ACS public use microdata sample for the state of North

Carolina. Drechsler and Hu (2021) generated partially synthetic data for large-scale administrative data containing detailed geographic information in Germany. Hu and Savitsky (2018) also used the DPMPM to generate partially synthetic data for the Consumer Expenditure Surveys (CE) at the U.S. Bureau of Labor Statistics (BLS), disseminating detailed county-level geographic information.

Among the work on synthetic data generation using the DPMPM with structural zeros, Manrique-Vallier and Hu (2018) proposed a data augmentation approach and generated fully synthetic data of repeated samples from the 5% public use microdata from the 2000 United States Census for the state of California, a population of 1,690,642 records measured in 17 categorical variables, with the number of levels ranging from 2 to 11.

Two ACS samples for illustrations

Before presenting detailed step-by-step illustrations to use the **NPBayesImputeCat** package for multiple imputation and data synthesis applications, we introduce two samples from the 2016 1-year American Community Surveys (ACS), which will both be used for our illustrations.

ACS sample 1, 'ss16pusa_sample_zeros', contains structural zeros. It will be used to illustrate how to perform multiple imputation and data synthesis tasks when structural zeros are present. ACS sample 2, 'ss16pusa_sample_nozeros', is a subset of ACS sample 1 and contains no structural zeros. It will be used to illustrate how to perform multiple imputation and data synthesis tasks when structural zeros are not present.

ACS sample 1, with structural zeros

Variable	Description	#	Category details
AGEP	Age	7	16; 17; [18, 24]; [25, 35]; [36, 50]; [51, 70]; (70,)
MAR	Marital status	5	Married; Widowed; Divorced; Separated; Never married.
SCHL	Education attainment	9	Up to K0; Some K12, no diploma; High school diploma or GED; Some college, no degree; Associate's degree; Bachelor's degree; Master's degree; Professional degree; Doctorate degree.
SEX	Sex	2	Male; Female
WKL	When last worked	3	Within the last 12 months; 1-5 years ago; Over 5 years ago or never worked.

Table 1: Variables used in ACS sample 1. The four table columns provide information on: variable name, simple description of the variable, the number of categories, and the details of the categories.

ACS sample 1 is a random sample of $n = 1,000$ observations on $p = 5$ variables. See Table 1 for the data dictionary. The sample is saved as 'ss16pusa_sample_zeros', and it contains structural zeros: 8 combinations, all related to AGEP and SCHL variables, listed in Table 2. These 8 cases are derived from the original 2016 1-year ACS data (as the population).

#	Description
1	AGEP = 16 & SCHL = Bachelor's degree.
2	AGEP = 16 & SCHL = Doctorate degree.
3	AGEP = 16 & SCHL = Master's degree.
4	AGEP = 16 & SCHL = Professional degree.
5	AGEP = 17 & SCHL = Bachelor's degree.
6	AGEP = 17 & SCHL = Doctorate degree.
7	AGEP = 17 & SCHL = Master's degree.
8	AGEP = 17 & SCHL = Professional degree.

Table 2: 8 cases of structural zeros in the ACS sample. The table columns include the index of each structural zeros case and simple description of the case itself.

ACS sample 2, without structural zeros

To obtain a sample without structural zeros, we take a subset of ACS sample 1, where $n = 1,000$ and $p = 3$, dropping variables AGEP and SCHL to eliminate any structural zeros. This ACS sample 2 is saved as 'ss16pusa_sample_nozeros'. See Table 3 for the data dictionary.

Variable	Description	#	Category details
MAR	Marital status	5	Married; Widowed; Divorced; Separated; Never married.
SEX	Sex	2	Male; Female
WKL	When last worked	3	Within the last 12 months; 1-5 years ago; Over 5 years ago or never worked.

Table 3: Variables used in ACS sample 2. The four table columns provide information on: variable name, simple description of the variable, the number of categories, and the details of the categories.

Missing data applications

To illustrate the applications of the **NPBayesImputeCat** package to missing data, we introduce 30% missingness for each variable in the ACS sample 1 and ACS sample 2 datasets, under the missing completely at random (MCAR) mechanism. The corresponding datasets (data containing missing values) to ACS sample 1 and ACS sample 2 are saved as ‘ss16pusa_sample_zeros_miss’ and ‘ss16pusa_sample_nozeros_miss’, respectively. The DPMPM imputation engine is designed to perform multiple imputations of categorical data that are missing at random (MAR)–and thus also data that are missing completely at random (MCAR).

Multiple imputation for data without structural zeros

We begin with the imputation of the missing values in the ACS sample 2 with 30% missingness, ‘ss16pusa_sample_nozeros_miss’, where no structural zeros are present. In the next section, we demonstrate how to impute missing values for ACS sample 1 with 30% missingness, ‘ss16pusa_sample_zeros_miss’, where structural zeros are present.

For each sample, we also compare the performance of the DPMPM engine to the most popular multiple imputation method, MICE. We implement the latter using the **mice** package in R. A brief review of **mice** is included at the beginning of the paper.

Load the sample data

First, we load the sample data, the ACS sample 2 with 30% missingness, and make sure that all variables are unordered factors.

```
data("ss16pusa_sample_nozeros_miss")
X <- ss16pusa_sample_nozeros_miss
p <- ncol(X)
for (j in 1:p){
  X[,j] <- as.factor(X[,j])
}
```

Initialize the DPMPM imputation engine

We use the `DPMPM_nozeros_imp` function to implement the DPMPM imputation engine without structural zeros. We first review the process for creating and initializing the DPMPM model using the `CreateModel` function to enable analysts to tune the number of mixture components through initial runs, before generating imputations using `DPMPM_nozeros_imp`. `CreateModel` is a wrapper function for creating an object of type “Lcm”. Lcm was implemented as an Rcpp module to expose the C++ implementation for our algorithm. Users can learn more about the `Lcm` class by typing `?`Lcm'`, which will bring up the R documentation for this class, including all methods and properties.

`CreateModel` takes 7 arguments as input:

1. `X`, the original data with missing values.
2. `MCZ`, the data frame containing the structural zeros definitions - use `NULL` when structural zeros are not present.
3. `K`, the maximum number of mixture components (i.e., the maximum number of latent classes in the DPMPM imputation engine).
4. `Nmax`, an upper truncation limit for the augmented sample size, that is, the maximum number of observations allowable in the augmented $\mathcal{X}^0$ - use `0` when structural zeros are not present.
5. `aalpha`, the hyper parameter a_α in stick-breaking prior distribution in Equation (6).

6. balpha, the hyper parameter b_α in stick-breaking prior distribution in Equation (6).
7. seed, the seed value.

As a quick demonstration, we let K = 30, aalpha = balpha = 0.25, and seed = 456. The code below creates and initializes the DPMPM imputation engine without structural zeros for the data stored in X.

```
model <- CreateModel(X = X,
                     MCZ = NULL,
                     K = 30,
                     Nmax = 0,
                     aalpha = 0.25,
                     balpha = 0.25,
                     seed = 456)
```

Next, we run the model object for a set of user-specified numbers of burn-ins, MCMC iterations, and thinning. For example, to run the MCMC sampler for 5 iterations post 2 burn-ins, thin every 1 iteration, and print the output at each iteration, run the following code.

```
> model$Run(burnin = 2,
            iter = 5,
            thinning = 1,
            silent = FALSE)
Initializing...
Run model without structural zeros.
iter = 0  kstar = 30 alpha = 1 Nmis = 0
iter = 0  kstar = 30 alpha = 7.81552 Nmis = 0
iter = 1  kstar = 30 alpha = 6.6941 Nmis = 0
iter = 2  kstar = 30 alpha = 4.60622 Nmis = 0
iter = 3  kstar = 30 alpha = 5.67409 Nmis = 0
```

Here, we show the first few lines of the output. The output prints out the iteration index as iter, the value of occupied mixture components or latent classes as kstar, posterior estimates of α (the concentration parameter in stick-breaking prior distribution in Equation (6)) as alpha, and the size of the augmented sample as Nmis. In our demonstration, Nmis is always 0 as the size of the augmented sample is 0 when there are no structural zeros.

It is important to keep track of the value of kstar as the **NPBayesImputeCat** package uses the truncated stick-breaking representation of the DP prior (Sethuraman, 1994). If the value of kstar is always K, the maximum number of mixture components, we should re-run the DPMPM model by specifying a larger value of K to allow a large enough number of mixture components to cluster the observations. For additional details on setting K, see Hu et al. (2014) and Akande et al. (2017).

The above initial run seems to suggest that the estimation uses almost all latent classes (kstar is close or is 30, which is what the maximum number of latent classes K set to). It is therefore prudent to increase the value of K when executing the CreateModel command, for example:

```
> model <- CreateModel(X = X,
                       MCZ = NULL,
                       K = 80,
                       Nmax = 0,
                       aalpha = 0.25,
                       balpha = 0.25,
                       seed = 456)
> model$Run(burnin = 2,
            iter = 5,
            thinning = 1,
            silent = FALSE)
Initializing...
Run model without structural zeros.
iter = 0  kstar = 80 alpha = 1 Nmis = 0
iter = 0  kstar = 78 alpha = 16.4979 Nmis = 0
iter = 1  kstar = 77 alpha = 17.281 Nmis = 0
iter = 2  kstar = 75 alpha = 24.4488 Nmis = 0
iter = 3  kstar = 79 alpha = 26.1196 Nmis = 0
```

Again, we only show the first few lines of the output. This time, setting K equal to 80 seems sufficiently large. Users should keep track of the value of kstar for the entire run and adjust K accordingly.

To diagnose convergence of parameters in the Gibbs sampler, one can use the EnableTracer option before running the sampler to track certain parameters. Examples of keeping posterior samples of alpha and kstar to access convergence are included in the accompanying R file.

Generate the imputed datasets

After setting K based on the initial runs, we now run the DPMPM imputation engine without structural zeros to create m imputed datasets. The function DPMPM_nozeros_imp takes 10 arguments as input:

1. X, the original data with missing values.
2. nrun, the number of MCMC iterations.
3. burn, the number of burn-in.
4. thin, the number of thinning.
5. K, the maximum number of mixture components (i.e., the maximum number of latent classes in the DPMPM imputation engine)
6. aalpha, the hyper parameter a_α in stick-breaking prior distribution in Equation (6).
7. balpha, the hyper parameter b_α in stick-breaking prior distribution in Equation (6).
8. m, the number of imputations.
9. seed, the seed value.
10. silent, default to TRUE. Set this parameter to FALSE if more iteration info are to be printed.

The output of DPMPM_nozeros_imp is a list containing:

1. impdata, the list of m imputed datasets.
2. origdata, the original data X.
3. alpha, the saved posterior draws of α, which can be used to check MCMC convergence.
4. kstar, the saved numbers of occupied mixture components, which can be used to check MCMC convergence and track whether the upper bound K is set large enough.

To run the DPMPM_nozeros_imp function to impute missing data for ACS sample 2 with 30% missingness, we run the code below. For this demonstration, we set nrun to 10000, burn to 5000, thin to 50, K to 80, both aalpha and balpha 0.25, and m to 10. Finally, we set the seed to 211.

```
m <- 10
Imp_DPMPM <- DPMPM_nozeros_imp(X = X,
                               nrun = 10000,
                               burn = 5000,
                               thin = 50,
                               K = 80,
                               aalpha = 0.25,
                               balpha = 0.25,
                               m = m,
                               seed = 211,
                               silent = TRUE)
```

The printed output from each iteration are omitted here. For a quick diagnostic check on whether the upper bound K is set large enough, we can use the kstar_MCMCdiag function which takes the following input arguments:

1. kstar, the vector output of kstar from running the DPMPM model.
2. nrun, the number of MCMC iterations used in running the DPMPM model.
3. burn, the number of burn-in iterations used in running the DPMPM model.
4. thin, the number of thinning used in running the DPMPM model.

Its output a list of two MCMC diagnostics figures:

1. Traceplot, the traceplot of kstar post burn-in and thinning.
2. Autocorrplot, the autocorrelation plot of kstar post burn-in and thinning.

We first load the **bayesplot** package before using the kstar_MCMCdiag function.

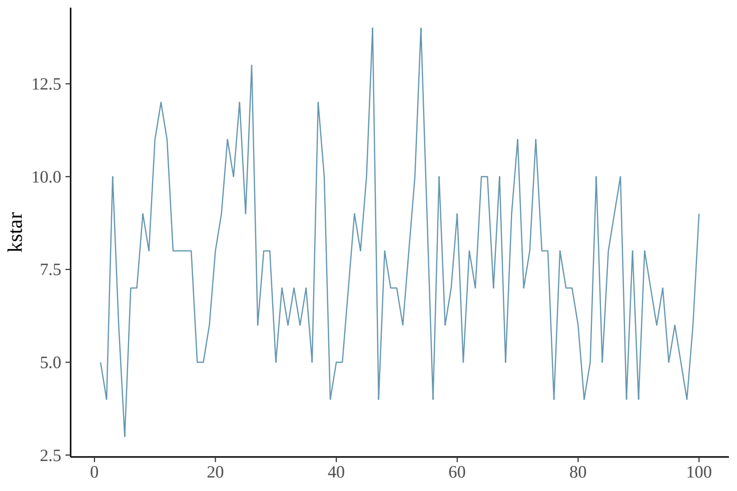


Figure 1: Traceplot of the thinned kstar values after burn-in. It shows little stickiness (only a 100 samples). The mean of kstar is 7.5, with range from 2.5 to 15. This suggests that setting $K = 80$ should be sufficient, and we could consider setting a smaller K for an even faster computation time.

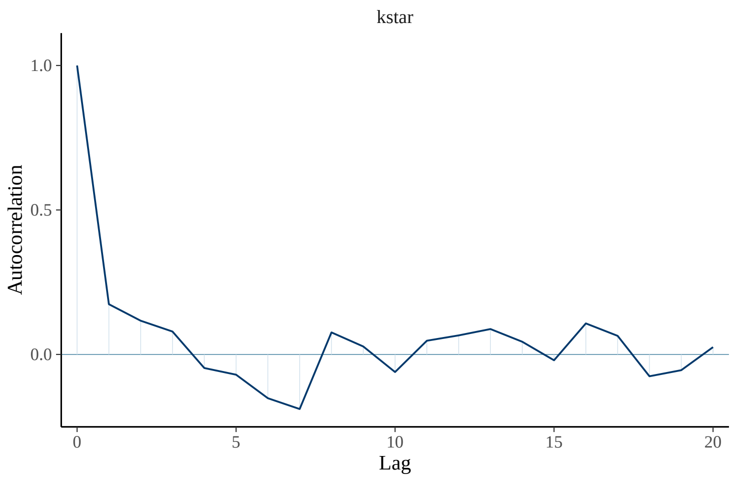


Figure 2: Autocorrelation plot of the thinned kstar values after burn-in. It shows a sharp drop in autocorrelation after lag 1, indicating no convergence issues after performing such MCMC diagnostics.

```
library(bayesplot)
kstar_MCMCdiag(kstar = Imp_DPMPM$kstar,
               nrun = 10000,
               burn = 5000,
               thin = 50)
```

Figure 1 shows the traceplot of `kstar` value after burn-in and thinning. It indicates no convergence issues of the MCMC chain. Moreover, it suggests choosing a smaller `K` value if we want to achieve an even faster computation time. Figure 2 presents its autocorrelation function plot, which also indicates no convergence issues.

To access the imputed datasets one at a time, we do the following.

```
impdata1 <- Imp_DPMPM$impdata[[1]] #for the first imputed dataset
```

Analysts then can compute sample estimates for estimands of interest in each imputed dataset and combine them using the combining rules.

Before demonstrating how to do so, we first use the **mice** package to also generate imputations for the same dataset. We do so to facilitate comparisons between results based on the DPMPM model and MICE. The following code runs the MICE algorithm on the same data using the default options in MICE for all the arguments, except `m`, which is set to 10 to be consistent with the implementations of the DPMPM engine. The code also reshapes the output of the MICE algorithm so that we are able to use some of the utility functions in **NPBayesImputeCat** on the imputed datasets. For more details on the implementations of **mice**, see Buuren and Groothuis-Oudshoorn (2011).

```
library(mice)
m <- 10
Imp_MICE <- mice(data = X,
                 m = m,
                 defaultMethod = c("norm", "logreg", "polyreg", "polr"),
                 print = F,
                 seed = 342)

#Reshape the list of imputed datasets
Imp_MICE_reshape <- NULL
Imp_MICE_reshape$impdata <- lapply(1:m,function(x) x <- X)
col_names <- names(Imp_MICE$imp)
for (l in 1:m){
  for(j in col_names){
    na_index_j <- which(is.na((Imp_MICE_reshape$impdata[[l]])[,j])==TRUE)
    Imp_MICE_reshape$impdata[[l]][na_index_j,j] <- Imp_MICE$imp[[j]][[l]]
  }
}
```

With the `Imp_DPMPM`, `Imp_MICE`, and `Imp_MICE_reshape` objects, we now demonstrate how to assess the quality of the imputations for the two methods and also use the combining rules for valid inferences from multiple imputed datasets.

Assess quality of the imputations

A very common way to assess the quality of the imputations is to compare the estimated distributions in the observed and imputed datasets. We can compare the marginal distributions of any of the variables in the observed and imputed datasets using the `marginal_compare_all_imp` function. The function takes 3 arguments as input:

1. `obsdata`, the observed data.
2. `impdata`, the list of m imputed datasets.
3. `vars`, the variable of interest.

The output is a list containing:

1. `Plot`, a barplot showing the marginal probability (as a percentage) of each level of the variable in the observed and imputed datasets.
2. `Comparison`, the table of the marginal probabilities (as a percentage) used to make the barplot.

As an example, we can compare the marginal probability of each level of the variable WKL in the observed and imputed datasets for both MICE and the DPMPM engine by using the following code. We load the **tidyverse** library for making these plots.

```
library(tidyverse)
marginal_compare_all_imp(obsdata = X,
                         impdata = Imp_DPMPM$impdata,
                         vars = "WKL")
marginal_compare_all_imp(obsdata = X,
                         impdata = Imp_MICE_reshape$impdata,
                         vars = "WKL")
```

The code creates the plots in Figures 3 and 4. For the most part in Figures 3 and 4, both DPMPM and MICE result in point estimates from the imputed datasets that are very close to the observed data, which are to be expected under MCAR. There are no major noticeable differences between the two methods. The code can be applied in a similar manner to other variables, MAR and SEX, shown in the accompanying R file.

Using the multiple imputation combining rules

We now demonstrate how to use the combining rules to obtain single point estimates and corresponding 95% confidence intervals for estimands of interest from all the imputed datasets. First, we compute

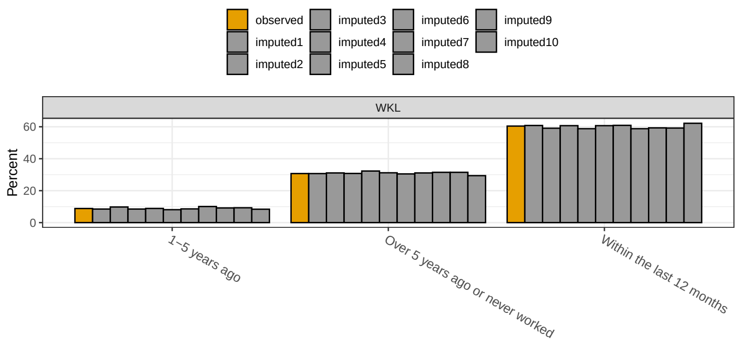


Figure 3: Marginal distribution of WKL from the observed data and each imputed dataset using DPMPM. Barplots of the observed (yellow) and the 10 imputed (grey) are shown for the three levels of WKL. There is some variability across the imputed datasets. Overall, they resemble the observed well.

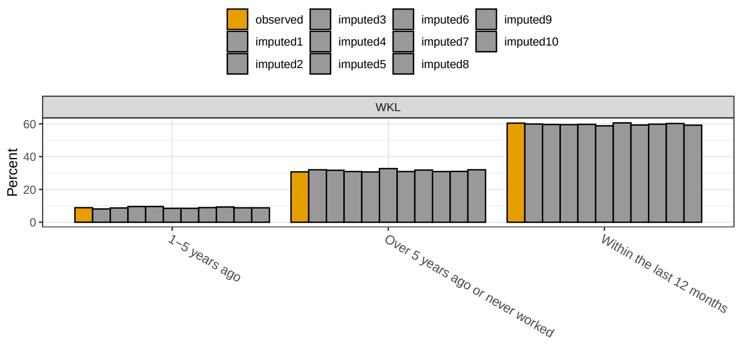


Figure 4: Marginal distribution of WKL from the observed data and each imputed dataset using MICE. Barplots of the observed (yellow) and the 10 imputed (grey) are shown for the three levels of WKL. There is some variability across the imputed datasets. Overall, they resemble the observed well.

the point estimates and corresponding standard errors for marginal and joint probabilities from each imputed dataset using the `compute_probs` function. The function takes 2 arguments as input:

1. `InputData`, the list of m imputed datasets.
2. `varlist`, a list of variable names (or combination of names) of interest.

The output is a list of the marginal and/or joint probabilities in each imputed dataset. Next, we use the `pool_estimated_probs` function to pool the estimates from all the imputed datasets using the combining rules. The function takes 2 arguments as input:

1. `ComputeProbsResults`, the output from the `compute_probs` function.
2. `method`, the combining rules to use, where the options are `"imputation"`, `"synthesis_full"`, `"synthesis_partial"`.

The output is a list of tables containing the results after applying the combining rules. For example, suppose we are interested in estimating probabilities corresponding to (i) the marginal distribution of MAR, (ii) the marginal distribution of SEX, and (iii) the joint distribution of MAR and WKL, we can use the following code.

```
varlist <- list(c("MAR"),c("SEX"),c("MAR","WKL")) #probabilities to evaluate
prob_ex1_DPMPM <- compute_probs(InputData = Imp_DPMPM$impdata,
                                varlist = varlist)
pooledprob_ex1_DPMPM <- pool_estimated_probs(ComputeProbsResults = prob_ex1_DPMPM,
                                             method = "imputation")

prob_ex1_MICE <- compute_probs(InputData = Imp_MICE_reshape$impdata,
                               varlist = varlist)
pooledprob_ex1_MICE <- pool_estimated_probs(ComputeProbsResults = prob_ex1_MICE,
                                            method = "imputation")
```

When dealing with missing data imputation, the `method` must be set to "`imputation`". The first element of `pooledprob_ex1_DPMPM` for MAR is shown below, whereas the remaining output is omitted for brevity.

```
                        MAR Estimate   Std.Error       Df Statistic   CI_Lower   CI_Upper
1                  Divorced   0.1083 0.011865562 9.000000  9.127254 0.08145823 0.13514177
2                   Married   0.5125 0.020191254 9.000001 25.382277 0.46682421 0.55817579
3 Never married or age<15   0.2953 0.018326210 9.000000 16.113534 0.25384323 0.33675677
4                 Separated   0.0204 0.005678460 9.000000  3.592523 0.00755443 0.03324557
5                   Widowed   0.0635 0.008683588 9.000000  7.312645 0.04385636 0.08314364
```

Each row represents the different levels of the corresponding variable(s). From left to right, the columns give the variable names and levels, the overall point estimates averaged across all imputed datasets, and the corresponding standard errors, degrees of freedom, test statistics, and confidence intervals. Similarly, for `pooledprob_ex1_MICE`, we have

```
                        MAR Estimate   Std.Error Df Statistic    CI_Lower   CI_Upper
1                  Divorced   0.1055 0.011501682  9  9.172571 0.079481387 0.13151861
2                   Married   0.5204 0.017156032  9 30.333355 0.481590360 0.55920964
3 Never married or age<15   0.2912 0.015438579  9 18.861839 0.256275507 0.32612449
4                 Separated   0.0175 0.005031389  9  3.478165 0.006118207 0.02888179
5                   Widowed   0.0654 0.009065137  9  7.214452 0.044893235 0.08590676
```

As the output shows, the results from both MICE and DPMPM are once again similar when looking at marginal probabilities of MAR, and both are indeed close to the results from the original sample without any missing data (which are excluded for brevity).

The **NPBayesImputeCat** package also includes similar functions, `fit_GLMs` and `pool_fitted_GLMs`, for fitting generalized linear models (GLMs) to each imputed datasets and pooling the results across all the datasets. The `fit_GLMs` function takes 2 arguments as input:

1. `InputData`, the list of `m` imputed datasets.
2. `exp`, the GLM expression for the model of interest (for `nnet` which must be loaded first).

The output is a list containing the estimated parameters from the GLM model fitted to each imputed dataset. The `pool_fitted_GLMs` pools the GLM estimates from all the imputed datasets using the combining rules. The function takes 2 arguments as input:

1. `GLMResults`, the output from the `fit_GLMs` function.
2. `method`, the combining rules to use, where the options are "`imputation`", "`synthesis_full`", "`synthesis_partial`".

For example, to fit a multinomial logistic model of MAR on SEX, we can use the following code.

```
library(nnet)
model_ex1_DPMPM <- fit_GLMs(InputData = Imp_DPMPM$impdata,
                            exp = multinom(formula = MAR~SEX))
pool_fitted_GLMs(GLMResults = model_ex1_DPMPM,
                 method = "imputation")
```

The second line yields the following output, with the numbers rounded up to 4 decimal places.

```
                   Levels   Parameter Estimate Std.Error      Df Statistic CI_Lower CI_Upper
1                 Married (Intercept)   1.5431    0.1783  9.0033    8.6536   1.1398   1.9465
2                 Married     SEXMale   0.0282    0.2597  9.0150    0.1085  -0.5591   0.6155
3 Never married or age<15 (Intercept)   0.8719    0.1849  9.0034    4.7145   0.4536   1.2902
4 Never married or age<15     SEXMale   0.2626    0.2632  9.0139    0.9976  -0.3327   0.8578
5               Separated (Intercept)  -1.6225    0.4380  9.1346   -3.7044  -2.6111  -0.6339
6               Separated     SEXMale  -0.2124    0.7787 10.3909   -0.2728  -1.9386   1.5138
7                 Widowed (Intercept)  -0.0666    0.2174  9.0059   -0.3062  -0.5584   0.4252
8                 Widowed     SEXMale  -1.5906    0.4781  9.1614   -3.3265  -2.6693  -0.5118
```

The `fit_GLMs` and `pool_fitted_GLMs` functions perform a similar role to the `with` and `pool` functions in the **mice** package. To fit the same model under MICE, we use the following code.

```
model_ex1_MICE <- with(data = Imp_MICE,
                       exp = multinom(formula = MAR~SEX))
summary(pool(model_ex1_MICE))
```

The second line yields the following output, with the numbers rounded up to 4 decimal places.

```
                 y.level        term estimate std.error  statistic      df  p.value
1                Married (Intercept)   1.6828    0.1835     9.1720 78.1913   0.0000
2                Married     SEXMale  -0.1735    0.2745    -0.6321 53.7652   0.5300
3 Never married or age<15 (Intercept)   0.8643    0.1962     4.4051 95.0509   0.0000
4 Never married or age<15     SEXMale   0.2828    0.2980     0.9491 48.8560   0.3473
5              Separated (Intercept)  -1.5754    0.6032    -2.6117 19.3114   0.0170
6              Separated     SEXMale  -0.7507    1.0821    -0.6937 17.6531   0.4969
7                Widowed (Intercept)   0.1187    0.2481     0.4783 49.8076   0.6345
8                Widowed     SEXMale  -2.2404    0.6741    -3.3235 32.0590   0.0022
```

The results are mostly similar, although we note that for most of the estimands, the standard errors are larger for MICE than the DPMPM engine. However, we also note that this illustration is based on data containing only $n = 1000$ observations and 30% missing data, so that differences in point estimates are not particularly surprising. Additional examples of fitting GLM models to the imputed datasets are shown in the accompanying R file.

Multiple imputation for data with structural zeros

We now illustrate how to impute missing values for ACS sample 1 with 30% missingness, using **NPBayesImputeCat**, where there are structural zeros are present. Recall that the data is stored in the file, 'ss16pusa_sample_zeros_miss'. The general procedure is very similar to the one in the previous section, where structural zeros are not present. However, we need to specify additional inputs to account for the structural zeros when generating the imputed datasets. Once the imputed datasets have been created, the utility functions used to compute sample estimates and pool them using the combining rules are exactly the same as before. First, we begin by creating MCZ, the data frame containing the structural zeros definition.

Create a file to store structural zeros cases

Previously, when there are no structural zeros, MCZ is set to NULL . Here, when there are structural zeros cases in the application, one should write the MCZ data frame following two general rules:

1. Variables in MCZ must be factors with the same levels as the original data.
2. Placeholder components are represented with NAs.

The script below is a sample script to store the structural zeros definition shown in Table 2.

```
AGEP <- c(16, 16, 16, 16, 17, 17, 17, 17)
SCHL <- c("Bachelor's degree", "Doctorate degree", "Master's degree",
          "Professional degree", "Bachelor's degree", "Doctorate degree",
          "Master's degree", "Professional degree")
MAR <- rep(NA, 8)
SEX <- rep(NA, 8)
WKL <- rep(NA, 8)
MCZ <- as.data.frame(cbind(AGEP, MAR, SCHL, SEX, WKL))
```

First, we create a vector of AGEP consisting of 4 replicates of value 16 and 4 replicates of value 17 and a vector of SCHL consisting of the degree types which induce structural zeros cases with AGEP. Second, we create vectors of MAR, SEX, and WKL. Each is a vector of length 8, with each element being NA. These are placeholder components, and since the structural zeros cases do not involve these three variables, all elements are NAs. Third, we need to create a data frame using as.data.frame and cbind. It is necessary to input the variables in the same order as in the original data (the order of variables in Table 1). We save the data frame MCZ for later use.

Load the sample

Now, we load the sample data and make sure that all variables are unordered factors.

```
data("ss16pusa_sample_zeros_miss")
X <- ss16pusa_sample_zeros_miss
```

```
p <- ncol(X)
for (j in 1:p){
  X[,j] <- as.factor(X[,j])
  MCZ[,j] <- factor(MCZ[,j], levels = levels(X[,j]))
}
```

Generate the imputed datasets

Initializing the DPMPM engine follows the exact same approach as before. The only difference is that we can now supply the two arguments specific to structural zeros, that is, `MCZ` and `Nmax`. That is, to initialize, we can run the following code.

```
model <- CreateModel(X = X,
                     MCZ = MCZ,
                     K = 30,
                     Nmax = 20000,
                     aalpha = 0.25,
                     balpha = 0.25,
                     seed = 521)
```

As before, we select `K` based on initial runs. We now also do the same for `Nmax`. If the value of either always hits the set values, we should re-run the model by specifying larger values to allow for a large enough number of mixture components and augmented observations to cluster the observations appropriately. As before, we can also save and track posterior samples of the parameters in the sampler using the `EnableTracer` option. Sample scripts are included in the accompanying R file.

After setting `K` and `Nmax`, we can now run the DPMPM imputation engine with structural zeros to create m imputed datasets. The function `DPMPM_zeros_imp` takes 12 arguments as input:

1. `X`, the original data with missing values.
2. `MCZ`, data frame containing the structural zeros definition.
3. `Nmax`, an upper truncation limit for the augmented sample size.
4. `nrun`, the number of MCMC iterations.
5. `burn`, the number of burn-in.
6. `thin`, the number of thinning.
7. `K`, the maximum number of mixture components (i.e., the maximum number of latent classes in the DPMPM imputation engine)
8. `aalpha`, the hyper parameter a_α in stick-breaking prior distribution in Equation (6).
9. `balpha`, the hyper parameter b_α in stick-breaking prior distribution in Equation (6).
10. `m`, the number of imputations.
11. `seed`, the seed value.
12. `silent`, default to TRUE. Set this parameter to FALSE if more iteration info are to be printed.

The output of `DPMPM_zeros_imp` is similar to the output of `DPMPM_nozeros_imp`, except that now it also includes `Nmis`, the saved posterior draws of the augmented sample size, which can be used to check MCMC convergence.

To run the `DPMPM_zeros_imp` function to impute missing data for ACS sample 1 with 30% missingness, we run the code below. For this demonstration, we set `Nmax` to 200000, `nrun` to 10000, `burn` to 5000, `thin` to 50, `K` to 80, both `aalpha` and `balpha` 0.25, and `m` to 10. Finally, we set the `seed` to 653.

```
m <- 10
Imp_DPMPM <- DPMPM_zeros_imp(X = X,
                             MCZ = MCZ,
                             Nmax = 200000,
                             nrun = 10000,
                             burn = 5000,
                             thin = 50,
                             K = 80,
                             aalpha = 0.25,
                             balpha = 0.25,
                             m = m,
                             seed = 653,
                             silent = TRUE)
```

As before, it is straightforward to run MCMC diagnostics on the tracked elements of `Imp_DPMPM`. Also, `Imp_DPMPM` contains the list of the imputed datasets. Analysts then can compute sample estimates for estimands of interest in each imputed dataset and assess their quality or also combine them using the combining rules by using all the same functions as before. That is, using `marginal_compare_all_imp`, `compute_probs`, `pool_estimated_probs`, `fit_GLMs`, and `pool_fitted_GLMs`. Examples are included in the accompanying R file.

Currently, there are no direct options in the **mice** package to incorporate structural zeros. Therefore, we do not explore comparisons with the MICE engine for data containing structural zeros.

Synthetic data applications

Without loss of generality, suppose we want to generate partially synthetic datasets for the ACS sample 2 ('ss16pusa_sample_nozeros'), where no structural zeros are present. **NPBayesImputeCat** includes functionality to generate fully synthetic data as well, but we exclude its illustration for brevity. The **NPBayesImputeCat** package currently does not accommodate data synthesis with structural zeros.

We also implement a popular synthetic data generation method, CART (using the **synthpop** package in R), to ACS sample 2 and compare the results (Reiter, 2005) to DPMPM.

Load the sample data

First, we load the sample data, the ACS sample 2, and make sure that all variables are set as factors.

```
data(ss16pusa_sample_nozeros)
X <- ss16pusa_sample_nozeros
p <- ncol(X)
for (j in 1:p){
  X[,j] <- as.factor(X[,j])
}
```

Generate the synthetic datasets

Initializing the DPMPM synthesizer follows the exact same approach as before for the missing data imputation applications. We run the following code.

```
model <- CreateModel(X = X,
                     MCZ = NULL,
                     K = 80,
                     Nmax = 0,
                     aalpha = 0.25,
                     balpha = 0.25,
                     seed = 973)
```

After setting `K` based on the initial runs, we now run the DPMPM synthesizer without structural zeros to create `m` synthetic datasets. The function `DPMPM_nozeros_syn` takes 12 arguments as input:

1. `X`, the original data with missing values.
2. `dj`, the vector recording the number of categories of the variables.
3. `nrun`, the total number of MCMC iterations.
4. `burn`, the number of burn-ins.
5. `thin`, the number of thinnings.
6. `K`, the maximum number of mixture components.
7. `aalpha`, the hyper parameter a_α in stick-breaking prior distribution in Equation (6).
8. `balpha`, the hyper parameter b_α in stick-breaking prior distribution in Equation (6).
9. `m`, the number of synthetic datasets.
10. `vars`, the names of the variables to be synthesized.
11. `seed`, the seed value.
12. `silent`, default to TRUE. Set this parameter to FALSE if more iteration info are to be printed.

The output of `DPMPM_nozeros_syn` is a list containing:

1. `syndata`, the list of m synthetic datasets.
2. `origdata`, the original data `X`.
3. `alpha`, the saved draws of α, which can be used to check MCMC convergence.
4. `kstar`, the saved numbers of occupied mixture components, which can be used to check MCMC convergence and track whether the upper bound `K` is set large enough.

To run the `DPMPM_nozeros_syn` function to generate synthetic data for ACS sample 2, we run the code below. For this demonstration, we create partially synthetic data where marital status (MAR) and when last worked (WKL) are synthesized, and we set `nrun` to 10000, `burn` to 5000, `thin` to 50, `K` to 80, both `aalpha` and `balpha` to 0.25, and `seed` to 837. Recall that `dj` stores the vector of levels of the variables, which are 5 for MAR, 2 for SEX, and 3 for WKL.

```
dj <- c(5, 2, 3)
m <- 5
Syn_DPMPM <- DPMPM_nozeros_syn(X = X,
                               dj = dj,
                               nrun = 10000,
                               burn = 5000,
                               thin = 50,
                               K = 80,
                               aalpha = 0.25,
                               balpha = 0.25,
                               m = 5,
                               vars = c("MAR", "WKL"),
                               seed = 837,
                               silent = TRUE)
```

MCMC diagnostics can be run based on the tracked elements of `Syn_DPMPM`. To access the synthetic datasets one at a time, we do the following.

```
syndata3 <- Syn_DPMPM$syndata[[3]] #for the third synthetic dataset
```

Analysts then can compute sample estimates for estimands of interest in each synthetic dataset and combine them using the combining rules. For comparison, we use the **synthpop** package to generate synthetic data for the same dataset with CART synthesizer. The default synthesizer for **synthpop** is CART, and the `visit.sequence` input argument allows the users to indicate which variables to be synthesized. To match with what we have done with the **NPBayesImputeCat**, we set MAR and WKL for `visit.sequence` to generate partially synthetic data.

```
library(synthpop)
m <- 5
Syn_CART <- syn(data = X,
                m = 5,
                seed = 123,
                visit.sequence = c("MAR", "WKL"))
```

Next, we demonstrate how to access the utility of the synthetic datasets from the two methods and also use the combining rules.

Assess utility of the synthetic datasets

Similar to what we have introduced for assessing the quality of the imputations, for synthesis, we compare the marginal distributions of any of the variables in the observed and imputed datasets, using the `marginal_compare_all_syn` function. The function takes 3 arguments as input:

1. `obsdata`, the observed data.
2. `syndata`, the list of m synthetic datasets.
3. `vars`, the variable of interest.

The output is a list containing:

1. `Plot`, a barplot showing the marginal probability (as a percentage) of each level of the variable in the observed and synthetic datasets.
2. `Comparison`, the table of the marginal probabilities (as a percentage) used to make the barplot.

The following code compares the marginal probability of each level of MAR.

```
marginal_compare_all_syn(obsdata = X,
                         syndata = Syn_DPMPM$syndata,
                         vars = "MAR")
marginal_compare_all_syn(obsdata = X,
                         syndata = Syn_CART$syn,
                         vars = "MAR")
```

The code creates the plots in Figures 5 and 6.

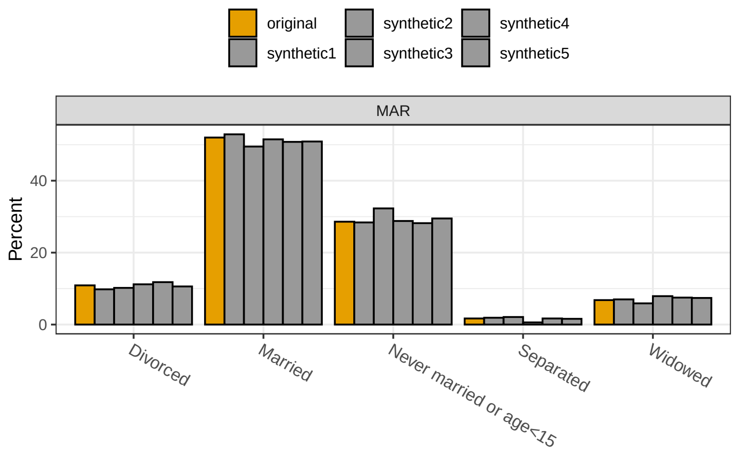


Figure 5: Marginal distribution of MAR from the observed data and each synthetic dataset using DPMPM. Barplots of the original (yellow) and the 5 synthetic (grey) are shown for the five levels of MAR. There is some variability across the synthetic datasets. Overall, they resemble the original well.

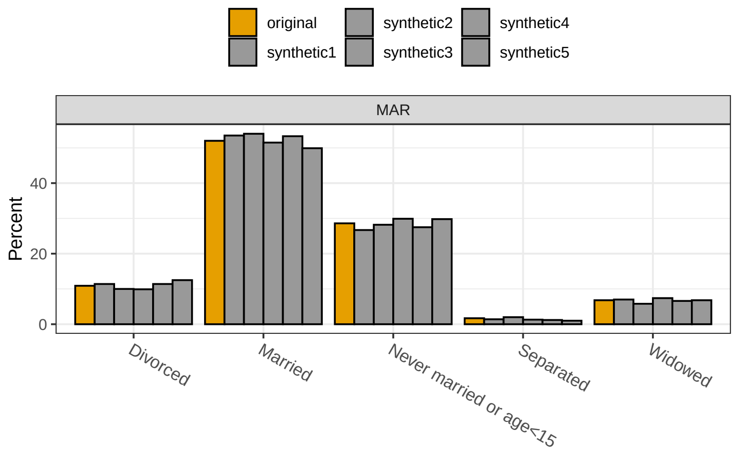


Figure 6: Marginal distribution of MAR from the observed data and each synthetic dataset using CART. Barplots of the original (yellow) and the 5 synthetic (grey) are shown for the five levels of MAR. There is some variability across the synthetic datasets. Overall, they resemble the original well.

For the most part in Figures 5 and 6, both DPMPM and CART result in point estimates from the synthetic datasets that are very close to the observed data. There are no major noticeable differences between the two methods. The code can be applied in a similar manner to other variables, and examples are included in the accompanying R file.

Using the synthetic data combining rules

We now demonstrate how to use the combining rules to obtain single point estimates and corresponding 95% confidence intervals for estimands of interest from all the synthetic datasets. Similar to what we have done with the multiple imputation combining rules previously, we use the same set of functions: `compute_probs` and `pool_estimated_prob`.

For example, suppose we are interested in estimating probabilities corresponding to (i) the marginal distribution of MAR, (ii) the marginal distribution of SEX, (iii) the marginal distribution of WKL, and (iv) the joint distribution of MAR and WKL, we can use the following code:

```
varlist <- list(c("MAR"), c("SEX"), c("WKL"), c("MAR","WKL")) #probabilities to evaluate
prob_ex1_DPMPM <- compute_probs(InputData = Syn_DPMPM$syndata,
                                varlist = varlist)
pooledprob_ex1_DPMPM <- pool_estimated_probs(ComputeProbsResults = prob_ex1_DPMPM,
                                             method = "synthesis_partial")

prob_ex1_CART <- compute_probs(InputData = Syn_CART$syn,
                               varlist = varlist)
pooledprob_ex1_CART <- pool_estimated_probs(ComputeProbsResults = prob_ex1_CART,
                                            method = "synthesis_partial")
```

As noted before, when dealing with partially synthetic data, the `method` must be set to "synthesis_partial". We only include the first element of `pooledprob_ex1_DPMPM` for MAR for brevity.

```
                       MAR Estimate  Std.Error        Df Statistic    CI_Lower   CI_Upper
1                 Divorced   0.1072 0.010406613 293.63208 10.301142 0.086718995 0.12768100
2                  Married   0.5112 0.016738961 338.91020 30.539531 0.478274660 0.54412534
3 Never married or age<15   0.2944 0.016234667  88.41528 18.134034 0.262139123 0.32666088
4                Separated   0.0158 0.004718411  43.64386  3.348585 0.006288476 0.02531152
5                  Widowed   0.0714 0.008826970 178.61153  8.088846 0.053981434 0.08881857
```

Each row represents the different levels of the corresponding variable(s). From left to right, the columns give the variable names and levels, the overall point estimates averaged across all imputed datasets, and the corresponding standard errors, degrees of freedom, test statistics, and confidence intervals. Similarly, for `pooledprob_ex1_CART`, we have

```
                       MAR Estimate   Std.Error       Df Statistic    CI_Lower   CI_Upper
1                 Divorced   0.1104 0.011044293 104.5372  9.996113 0.088500078 0.13229992
2                  Married   0.5244 0.017530887 111.6932 29.912919 0.489663753 0.55913625
3 Never married or age<15   0.2842 0.015588823 149.5745 18.231011 0.253397249 0.31500275
4                Separated   0.0138 0.004054405 134.0083  3.403705 0.005781098 0.02181890
5                  Widowed   0.0672 0.008348413 392.0400  8.049434 0.050786740 0.08361326
```

As the output shows, the results from both CART and DPMPM are once again similar when looking at marginal probabilities of MAR, and both are indeed close to the results from the original sample (excluded for brevity).

Lastly, we demonstrate the use of `fit_GLMs` and `pool_fitted_GLMs` for fitting generalized linear models (GLMs) to each synthetic datasets and pooling the results across all the datasets. For example, to fit a logistic model of SEX given MAR and WKL, we can use the following code.

```
model_ex1_DPMPM <- fit_GLMs(InputData = Syn_DPMPM$syndata,
                            exp = glm(formula = SEX~WKL+MAR,
                                      family = binomial))
pool_fitted_GLMs(GLMResults = model_ex1_DPMPM,
                 method = "synthesis_partial")
```

The second line yields the following output, with the numbers rounded up to 4 decimal places.

```
                              Estimate Std.Error       Df  Statistic   CI_Lower   CI_Upper
(Intercept)                    -0.5954 0.3548  44.1182 -1.6781 -1.3105  0.1196
WKLOver 5 years ago or never worked  0.0946 0.2823 179.2380  0.3351 -0.4625  0.6517
WKLWithin the last 12 months     0.3629 0.3042  33.3456  1.1929 -0.2558  0.9816
MARMarried                       0.3517 0.2581  48.6246  1.3625 -0.1671  0.8706
MARNever married or age<15       0.4293 0.2560 119.8199  1.6771 -0.0775  0.9360
MARSeparated                     0.0710 0.6589 545.7313  0.1078 -1.2232  1.3652
MARWidowed                      -1.0156 0.3864 543.1589 -2.6285 -1.7746 -0.2566
```

We use the following code to fit the same model under CART.

```
model_ex1_CART <- fit_GLMs(InputData = Syn_CART$syn,
                           exp = glm(formula = as.factor(SEX)~WKL+MAR,
                                     family = binomial))
pool_fitted_GLMs(GLMResults = model_ex1_CART,
                 method = "synthesis_partial")
```

The second line yields the following output, with the numbers rounded up to 4 decimal places.

```
                                Estimate  Std.Error     Df  Statistic  CI_Lower  CI_Upper
(Intercept)                          -0.0292 0.3155  103.8094 -0.0927 -0.6549 0.5964
WKLOver 5 years ago or never worked -0.0610 0.2941   56.6928 -0.2074 -0.6499 0.5279
WKLWithin the last 12 months        -0.1309 0.2602  125.4832 -0.5032 -0.6460 0.3841
MARMarried                           0.0316 0.2151 3096.9978  0.1471 -0.3902 0.4534
MARNever married or age<15           0.0230 0.2401  319.0640  0.0956 -0.4495 0.4954
MARSeparated                         0.3378 0.6530  187.2752  0.5172 -0.9505 1.6260
MARWidowed                           0.1148 0.3365  477.2065  0.3412 -0.5464 0.7760
```

The output of fitting this GLM to the original sample is used as the benchmark for our utility evaluation.

```
                                     Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -0.5937     0.2897  -2.050 0.040395 *
WKLOver 5 years ago or never worked    0.2859     0.2539   1.126 0.260163
WKLWithin the last 12 months           0.5054     0.2358   2.144 0.032065 *
MARMarried                             0.1380     0.2129   0.649 0.516647
MARNever married or age<15             0.3994     0.2273   1.757 0.078965 .
MARSeparated                          -0.3538     0.5463  -0.648 0.517183
MARWidowed                            -1.3673     0.3867  -3.536 0.000406 ***
```

The utility results are different between DPMPM and CART, and we note that for most of the estimands, the DPMPM produces more accurate estimation than the CART when compared to the results from the original sample. These indicate that for this particular sample, a joint model fitted by the DPMPM does a better job preserving relationships among variables compared to a series of the conditional models fitted by the CART. Additional examples of fitting GLM models to the synthetic datasets are shown in the accompanying R file.

Concluding remarks

In this paper, we have presented the DPMPM models for multivariate categorical data and illustrations of using the **NPBayesImputeCat** package for multiple imputation and synthetic data applications. Users can take the output and extract imputed and synthetic datasets, then conduct statistical analyses of their choice and use the appropriate combining rules to obtain valid estimates. Interested readers can refer to the package documentation for additional features.

While the **NPBayesImputeCat** package has been developed primarily for multiple imputation and synthetic data purposes, users can also use it for DPMPM model estimation. For example, following the illustrations for synthetic data, a data analyst is able to obtain parameter draws of several key parameters from the MCMC chain: i) the DP concentration parameter α, ii) the mixture probability vectors $\{\pi_k\}$, and iii) the Multinomial probability vectors $\{\theta_k^{(j)}\}$. The analyst can then further conduct analyses of the clustering of the observations in the MCMC chain and other questions of interest.

Users can report bugs at our GitHub repo: `https://github.com/monika76five/NPBayesImputeCat`.

Bibliography

O. Akande, F. Li, and J. P. Reiter. An empirical comparison of multiple imputation methods for categorical data. *The American Statistician*, 71(2):162–170, 2017. URL `https://doi.org/10.1080/00031305.2016.1277158`. [p92, 93, 96]

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Belmont, CA: Wadsworh, Inc., 1984. URL `https://www.routledge.com/Classification-and-Regression-Trees/Breiman-Friedman-Stone-Olshen/p/book/9780412048418`. [p90]

L. Burgette and J. P. Reiter. Multiple imputation via sequential regression trees. *American Journal of Epidemiology*, 172(9):1070–1076, 2010. URL https://doi.org/10.1093/aje/kwq260. [p90]

S. Buuren and C. Groothuis-Oudshoorn. Mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45, 12 2011. URL https://doi.org/10.18637/jss.v045.i03. [p90, 99]

J. Drechsler. *Synthetic datasets for statistical disclosure control*. Springer-Verlag, 2011. URL https://www.springer.com/gp/book/9781461403258. [p90, 91]

J. Drechsler and J. Hu. Synthesizing geocodes to facilitate access to detailed geographical information in large scale administrative data. *Journal of Statistics and Survey Methodology*, 9(3):523–548, 2021. URL https://doi.org/10.1093/jssam/smaa035. [p92, 94]

D. B. Dunson and C. Xing. Nonparametric bayes modeling of multivariate categorical data. *Journal of the American Statistical Association*, 104(487):1042–1051, 2009. URL https://doi.org/10.1198/jasa.2009.tm08439. [p92]

O. Harel and X. H. Zhou. Multiple imputation: review of theory, implementation and software. *Statistics in Medicine*, 26(16):3057–3077, 2007. URL https://doi.org/10.1002/sim.2787. [p90]

J. Hu and N. Hoshino. The Quasi-Multinomial synthesizer for categorical data. In J. Domingo-Ferrer and F. Montes, editors, *Privacy in Statistical Databases*, volume 11126 of *Lecture Notes in Computer Science*, pages 75–91. Springer, 2018. URL https://doi.org/10.1007/978-3-319-99771-1_6. [p92]

J. Hu and T. Savitsky. Bayesian data synthesis and disclosure risks quantification: an application to the consumer expenditure surveys. page arXiv:1809.10074, 2018. URL https://arxiv.org/abs/1809.10074. [p92, 94]

J. Hu, J. P. Reiter, and Q. Wang. Disclosure risk evaluation for fully synthetic categorical data. In J. Domingo-Ferrer, editor, *Privacy in Statistical Databases*, volume 8744 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2014. URL https://doi.org/10.1007/978-3-319-11257-2_15. [p92, 93, 96]

A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. S. Nordhold, K. Spicer, and P. de Wolf. *Statistical Disclosure Control*. Wiley Series in Survey Methodology, 2012. URL https://doi.org/10.1002/9781118348239. [p90]

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173, 2001. URL https://doi.org/10.1198/016214501750332758. [p92]

R. J. A. Little. Statistical analysis of masked data. *Journal of Official Statistics*, 9(2):407–426, 1993. [p90, 91]

D. Manrique-Vallier and J. Hu. Bayesian non-parametric generation of fully synthetic multivariate categorical data in the presence of structural zeros. *Journal of the Royal Statistical Society, Series A*, 181 (3):635–647, 2018. URL https://doi.org/10.1111/rssa.12352. [p92, 93, 94]

D. Manrique-Vallier and J. P. Reiter. Bayesian estimation of discrete multivariate latent structure models with structural zeros. *Journal of Computational and Graphical Statistics*, 23(4):1061–1079, 2014a. URL https://doi.org/10.1080/10618600.2013.844700. [p92, 93]

D. Manrique-Vallier and J. P. Reiter. Bayesian multiple imputation for large-scale categorical data with structural zeros. *Survey Methodology*, 40(1):125–134, 2014b. URL https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X201400114002. [p93]

J. S. Murray. Multiple imputation: a review of practical and theoretical findings. *Statistical Science*, 33 (2):142–149, 2018. URL https://doi.org/10.1214/18-STS644. [p93]

J. P. Reiter. Using CART to generate partially synthetic public use microdata. *Journal of Official Statistics*, 21:441–462, 2005. [p104]

J. P. Reiter and E. Raghunathan. The multiple adaptations of multiple imputation. *Journal of the American Statistical Society*, 102(480):1462–1471, 2007. URL https://doi.org/10.1198/016214507000000932. [p90]

D. B. Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, New York, NY, 1987. URL https://www.wiley.com/en-us/Multiple+Imputation+for+Nonresponse+in+Surveys-p-9780471655749. [p90]

D. B. Rubin. Discussion: Statistical disclosure limitation. *Journal of Official Statistics*, 9(2):461–468, 1993. [p90, 91]

D. B. Rubin. Multiple imputation after 18+ years. *Journal of the American Statistical Association*, 91(434): 473–489, 1996. [p90]

J. Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994. URL `www.jstor.org/stable/24305538`. [p92, 96]

Y. Si and J. P. Reiter. Nonparametric Bayesian multiple imputation for incomplete categorical variables in large-scale assessment surveys. *Journal of Educational and Behavioral Statistics*, 38:499–521, 2013. URL `https://doi.org/10.3102/1076998613480394`. [p92, 93]

Jingchen Hu
Mathematics and Statistics Department
Vassar College
Box 27, 124 Raymond Ave
Poughkeepsie, NY 12604, USA
ORCID: 0000-0002-4283-181X
E-mail: `jihu@vassar.edu`

Olanrewaju Akande
Social Science Research Institute, and
The Department of Statistical Science
Box 90989, Duke University
Durham, NC 27708, USA
ORCID: 0000-0002-0790-573X
E-mail: `olanrewaju.akande@duke.edu`

Quanli Wang
Department of Statistical Science
Box 90251, Duke University
Durham, NC 27708, USA
E-mail: `quanliwang20@gmail.com`

msae: An R Package of Multivariate Fay-Herriot Models for Small Area Estimation

by Novia Permatasari and Azka Ubaidillah

Abstract The paper introduces an R Package of multivariate Fay-Herriot models for small area estimation named **msae**. This package implements four types of Fay-Herriot models, including univariate Fay-Herriot model (model 0), multivariate Fay-Herriot model (model 1), autoregressive multivariate Fay-Herriot model (model 2), and heteroskedastic autoregressive multivariate Fay-Herriot model (model 3). It also contains some datasets generated based on multivariate Fay-Herriot models. We describe and implement functions through various practical examples. Multivariate Fay-Herriot models produce a more efficient parameter estimation than direct estimation and univariate model.

Introduction

Survey sampling is a data collection method by observing several units of observation to obtain information from the entire population. Compared to other data collection methods, survey sampling has advantages in cost, time, and human resources. Survey sampling is designed for a certain size of the domain, commonly a large area. However, data demand for small areas is increasing and has become high issue (Ghosh and Rao, 1994). The inadequate sample size causes a large standard error of parameter estimates. This problem is overcome by indirect estimation, namely Small Area Estimation (SAE).

Rao and Molina (2015) said that SAE increases the effectiveness of sample size using the strength of neighboring areas and information on other variables that are related to the variable of interest. There are some estimation methods in SAE, namely Best Linear Unbiased Predictors (BLUP), Empirical Best Linear Unbiased Predictors (EBLUP), Hierarchical Bayes (HB), and Empirical Bayes (EB). The most common SAE estimator is the EBLUP (Krieg et al., 2015). EBLUP has advantages over EB and HB methods. It is a development of the BLUP method that minimizes the MSE among other unbiased linear estimators (Ghosh and Rao, 1994). Area level of EBLUP application for the continuous response variable, called Fay-Herriot model, was firstly employed for estimating log per-capita income (PCI) in small places in the US (Rao and Molina, 2015).

The Fay-Herriot model has extended into a multivariate Fay-Herriot model, which is a model with several correlated response variables. Datta et al. (1991) firstly applied the multivariate model to estimate the median income of four-person families in the US states. Benavent and Morales (2016) developed multivariate Fay-Herriot models with the EBLUP method and introduced four estimation models based on the estimated variance matrix structure. Ubaidillah et al. (2019) also implemented the multivariate Fay-Herriot model and indicated that the multivariate Fay-Herriot model produces a more efficient parameter estimation than the univariate model.

On the Comprehensive R Archive Network (CRAN), there are several packages implementing small area estimation. Some of them are included in the Small Area Estimation subsection of The CRAN Task View: Official Statistics & Survey Methodology (Templ, 2014), including **sae** (Molina and Marhuenda, 2018), **rsae** (Schoch, 2014), **nlme** (Pinheiro et al., 2020), **hbsae** (Boonstra, 2012), **JoSAE** (Breidenbach, 2018), and **BayesSAE** (Chengchun Shi Developer, 2018). Other popular SAE packages not included in that subsection are **mme** (Lopez-Vizcaino et al., 2019) and **saery** (Lefler et al., 2014).

In this paper, we introduce our R package of multivariate Fay-Herriot models for small area estimation, named **msae**. This package and its details are available on CRAN at `http://CRAN.R-project.org/package=msae`. Functions in this package implement four Fay-Herriot Models, namely Model 0, Model 1, Model 2, and Model 3, as proposed by Benavent and Morales (2016).

The paper is structured as follows. First, we explain multivariate Fay-Herriot models in Section 2.2. Then, we describe **msae** package and illustrate the use of this package for SAE estimation by employing simulation studies and applying it to a real dataset in the next Sections 2.3 and 2.4. Finally, we provide a conclusion in Section 2.8.

Multivariate Fay-Herriot model

The multivariate Fay-Herriot model is an extension of the Fay-Herriot model, which utilizes some correlated responses. Fay-Herriot is a combination of two model components. The first component is called the sampling model, and the second component is called the linking model.

Suppose we want to estimate characteristics of R variables in D areas, $\mu_d = (\mu_{1d}, \ldots, \mu_{Rd})^T$, with $d = 1, \ldots, D$. Let $y_d = (y_{1d}, \ldots, y_{Rd})^T$, be a direct estimator of μ_d. The first component, i.e., sampling model is

$$y_d = \mu_d + e_d, \quad e_d \sim N(0, V_{e_d}), \quad d = 1, \ldots, D$$

, where e_d is sampling error with a covariance matrix, V_{e_d} , that is assumed to be known. In the second component, we assume that μ_d is linearly related with p_d area-specific auxiliary variables $X_d = (X_1, \ldots, X_{p_d})^T$ as follows:

$$\mu_d = X_d\beta_d + u_d, \quad u_d \sim N(0, V_{u_d}), \quad d = 1, \ldots, D$$

, where u_d is area random effects, and β_d is a vector of regression coefficient corresponding with X_d. This second component is called the linking model. The combination of the two components forms a multivariate linear mixed model as follows:

$$y_d = X_d\beta_d + u_d + e_d, \quad e_d \sim N(0, V_{e_d}), \quad d = 1, \ldots, D$$

, where u and e are independent.

Benavent and Morales (2016) proposed Fay-Herriot models using four different variance matrices, Model 0, Model 1, Model 2, and Model 3. Model 0 is a univariate Fay-Herriot Model, of which the sampling error and the random effect of target variables are independent. Sampling error and random effect variance matrix are written as follows:

$$V_{ud} = diag_{a \le r \le R}(\sigma^2_{ur})$$
$$V_{ed} = diag_{a \le r \le R}(\sigma^2_{edr})$$

,

$$\text{where } d = 1, \ldots, D$$

.

Model 1, Model 2, and Model 3 are multivariate Fay-Herriot models, of which the variance matrices are no longer diagonal matrices. Model 1 is a multivariate form of Model 0, where the random effect variance of Model 1 is still a diagonal matrix. Model 2 is called the autoregressive multivariate Fay-Herriot model (AR(1)), in which the random variance matrix is written as follows :

$$V_{ud} = \sigma^2_{ur}\Omega_d(\rho)$$

$$\Omega_d(\rho) = \frac{1}{1-\rho^2}\begin{bmatrix} 1 & \rho & \cdots & \rho^{R-1} \\ \rho & 1 & & \rho^{R-2} \\ \vdots & & \ddots & \vdots \\ \rho^{R-1} & \rho^{R-2} & \cdots & 1 \end{bmatrix}$$

Model 3 is called heteroskedastic autoregressive multivariate Fay-Herriot model (HAR(1)), which the element of random error is written as follows:

$$u_{dr} = \rho u_{dr-1} + a_{dr}$$
$$u_{d0} \sim N(0, \sigma^2_{u0}) \quad \text{and} \quad a_{dr} \sim N(0, \sigma^2_r)$$

, where $\sigma^2_{u0} = 1, a_{dr}$, and u_{d0} are independent. The element of random variance matrix is written as follow:

$$\sigma_{drii} = \sum_{k=1}^{i} \rho^{2k}\sigma^2_{i-k}$$

$$\sigma_{drij} = \sum_{k=0}^{|i-j|} \rho^{2k+|i-j|}\sigma^2_{|i-j|-k}$$

.

BLUP and EBLUP

The best linear unbiased prediction (BLUP) of μ is

$$\hat{\mu} = X\hat{\beta} + Z^T \hat{V}_u Z \Omega^{-1}(y - X\hat{\beta})$$

, where $\hat{\beta} = (X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}y$ is the best linear unbiased estimator (BLUE) of β with the covariance matrix $cov(\hat{\beta}) = (X^T\Omega^{-1}X)^{-1}$. BLUP estimator depends on the random effect variance that is usually unknown. Using Restricted Maximum Likelihood (REML), we estimate and substitute random effect variance estimator for obtaining the multivariate EBLUB estimator. The estimation formula with EBLUP is written as follows:

$$\hat{\mu} = X\hat{\beta} + Z^T \hat{V}_u Z \hat{\Omega}^{-1}(y - X\hat{\beta})$$

$$\hat{\Omega} = Z^T \hat{V}_u Z + V_e$$

, where $\hat{\beta} = (X^T\hat{\Omega}^{-1}X)^{-1}X^T\hat{\Omega}^{-1}y$ is the best linear unbiased estimator (BLUE) of β with the covariance matrix $cov(\hat{\beta}) = (X^T\hat{\Omega}^{-1}X)^{-1}$.

MSE

Benavent and Morales (2016) also proposed MSE estimation for the multivariate Fay-Herriot models as follows:

$$mse(\hat{\mu}) = g_{1i}(\hat{\sigma}_u^2) + g_{2i}(\hat{\sigma}_u^2) + g_{3i}(\hat{\sigma}_u^2)$$

, where each component can be described as follows:

$$g_{1i}(\hat{\sigma}_u^2) = \Gamma V_e$$

,

$$g_{2i}(\hat{\sigma}_u^2) = (1-\Gamma)X(X^T\Omega^{-1}X)^{-1}X^T(I-\Gamma)^T$$

,

$$g_{3i}(\hat{\sigma}_u^3) \approx \Sigma\Sigma cov(\hat{\sigma}_{uk}^2, \hat{\sigma}_{ul}^2)\Gamma_{(k)}\Omega\Gamma_{(k)}^T, k, l = 1, 2, \ldots, q$$

, where $\Gamma = Z^T\hat{V}_u Z$, $\Gamma_{(k)} = \frac{\delta\Gamma}{\delta\sigma_u^2}$, and $cov(\hat{\sigma}_{uk}^2, \hat{\sigma}_{ul}^2)$ is the inverse of the Fisher information matrix in the estimation of REML.

Overview of R package msae

The R Package **msae** implements multivariate Fay-Herriot models for small area estimation. Here are some functions and the descriptions at a glance:

- `eblupUFH`: This function gives the EBLUP and MSE based on the univariate Fay-Herriot model (Model 0).
- `eblupMFH1`: This function gives the EBLUP and MSE based on the multivariate Fay-Herriot model (Model 1).
- `eblupMFH2`: This function gives the EBLUP and MSE based on the autoregressive multivariate Fay-Herriot model (Model 2).
- `eblupMFH3`: This function gives the EBLUP and MSE based on the heteroskedastic autoregressive multivariate Fay-Herriot model (Model 3).

Those functions return a list of five elements:

- `eblup` is a data frame of EBLUPs for each variable.
- `MSE` is a data frame of the estimated MSEs of the EBLUPs.
- `randomEffect` is a data frame of random effect estimators.
- `Rmatrix` is a block diagonal matrix composed of sampling variances.
- `fit` is a list containing the following objects:
  - `method` shows the type of fitting method.
  - `convergence` shows the convergence of the Fisher Scoring algorithm.
  - `estcoef` shows estimated model coefficients and their significance.
  - `refvar` shows the estimated random effect variance.

- `refvarTest` (only for `eblupMFH3`) shows homogeneity of random effect variance test based on Model 3.
- `rho` (only for `eblupMFH2` and `eblupMFH3`) shows the estimated ρ of random effect variance and their parameter test.
- `informationFisher` is a matrix of information Fisher.

This package also provides datasets generated for each multivariate model. The datasets are generated based on Model 1, Model 2, and Model 3 following steps:

1. Generate sampling error e and auxiliary variables « $X1, X2$ ». Set the parameter as follows:
   - For sampling error e in Model 1, we set $e_d \sim N_3(0, V_{ed})$, where $V_{ed} = (\sigma_{dij})_{i,j=1,2,3}$ with $\sigma_{e11} \sim InvGamma(11,1)$, $\sigma_{e22} \sim InvGamma(11,2)$, $\sigma_{e33} \sim InvGamma(11,3)$, and $\rho_e = 0.5$. We generate different MSE of direct estimates for each area.
   - For sampling error e in Model 2 and Model 3, we set $e \sim N_3(0, V_e)$, where $V_e = (\sigma_{ij})_{i,j=1,2,3}$ with $\sigma_{e11} = 0.1$, $\sigma_{e22} = 0.2$, $\sigma_{e33} = 0.3$, and $\rho_e = 0.5$. It is shown that all the areas have the same MSE of direct estimates.
   - For auxiliary variables « $X1$ $X2$ », we set $X1 \sim N(5, 0.1)$ and $X2 \sim N(10, 0.2)$
2. Generate random effect u, where $u \sim N_3(0, V_u)$. For each dataset, parameter for generating random effect u are as follows:
   - For Model 1, $\sigma_{u11} = 0.2$, $\sigma_{u22} = 0.4$, and $\sigma_{u33} = 1.2$
   - For Model 2, $\sigma_u = 0.4$, and $\rho_u = 0.8$
   - For Model 3, $\sigma_{u11} = 0.2$, $\sigma_{u22} = 0.4$, $\sigma_{u33} = 1.2$, and $\rho_u = 0.8$
3. Set $\beta_1 = 5$ and $\beta_2 = 10$ to calculate direct estimation « $Y1$, $Y2$, and $Y3$ », where $Y_i = X\beta + u_i + e_i$.
4. Auxiliary variables « $X1$ and $X2$ », direct estimates « $Y1, Y2$, and $Y3$ », and sampling variance-covariance « $v1, v2, v3, v12, v13$, and $v23$ » are combined into a data frame called `datasae1` for Model 1, `datasae2` for Model 2, and `datasae3` for Model 3.

Example 1. The multivariate Fay-Herriot model (Model 1)

`datasae1`, which is generated based on Model 1, contains 50 observations on the following 11 variables: 3 dependent variables « $Y1, Y2$, and $Y3$ », 2 auxiliary variables « $X1$ and $X2$ », and 6 variance-covariance of direct estimation « $v1, v2, v3, v12, v13$, and $v23$ ». The procedures for generating such datasets are provided in the previous section. The following R commands are run to obtain EBLUPs of the univariate Fay-Herriot model (Model 0) and the multivariate Fay-Herriot model (Model 1), plot the EBLUPs of the univariate and multivarate model, and plot the MSEs of EBLUPs in the univariate and multivarate model:

```
data('datasae1')

# model specifications
Fo <- list(f1=Y1~X1+X2,
           f2=Y2~X1+X2,
           f3=Y3~X1+X2)
vardir <- c("v1", "v2", "v3", "v12", "v13", "v23")

# EBLUP based on Model 0 and Model 1
u <- eblupUFH(Fo, vardir, data=datasae1)  # Model 0
m1 <- eblupMFH1(Fo, vardir, data=datasae1)  #Model 1

# Figure 1: EBLUPs under Model 0 and Model 1
par(mfrow=c(1,3))

plot(u$eblup$Y1, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "Y1",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(m1$eblup$Y1, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
    pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)
```

```
plot(u$eblup$Y2, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "Y2",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(m1$eblup$Y2, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
    pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)
plot(u$eblup$Y3, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "Y3",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(m1$eblup$Y3, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
    pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)


# Figure 2: MSE of Model 0 and Model 1
par(mfrow=c(1,3))

plot(u$MSE$Y1, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "MSE of Y1",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n", ylim=c(0.038,0.12))
points(m1$MSE$Y1, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
       pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)
plot(u$MSE$Y2, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "MSE of Y2",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n", ylim=c(0.05,0.28))
points(m1$MSE$Y2, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
       pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)
plot(u$MSE$Y3, type = "o", col = "blue", pch = 15, xlab = "area", ylab = "MSE of Y3",
     cex.axis = 1.5, cex.lab = 1.5, xaxt = "n", ylim=c(0.1,0.42))
points(m1$MSE$Y3, type = "o", col = "red", pch = 18)
axis(1, at=1:50, labels = 1:50)
legend("topleft",legend=c("Model 0","Model 1"),ncol=2 , col=c("blue","red"),
       pch=c(15,18), inset=c(0.617,-0.1), xpd=TRUE)
```

Figure 1 illustrates the EBLUPs based on Model 0 (univariate model) and Model 1 (multivariate model) for all variables of interest. Figure 2 shows the MSEs of Model 1 compared with the MSEs of Model 0. It can be seen that the estimates of both methods show a similar pattern. Meanwhile, EBLUPs based on Model 1 has a lower MSE than Model 0. From this example, we can conclude that the multivariate Fay-Herriot model (Model 1) is more efficient than the univariate Fay-Herriot model (Model 0).

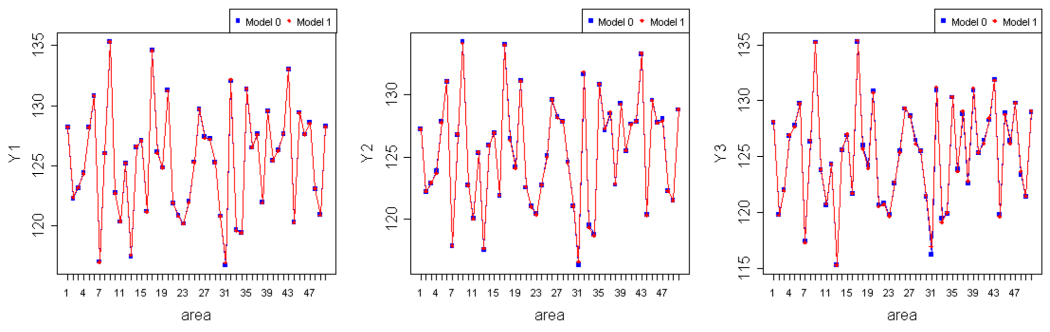


Figure 1: EBLUPs under Model 0 and Model 1.

Example 2. The autoregressive multivariate Fay-Herriot model (Model 2)

`datasae2`, which was generated based on Model 2, contains 50 observations on the following 11 variables: 3 dependent variables « $Y1, Y2$ and $Y3$ », 2 auxiliary variables « $X1$ and $X2$ », and 6 variance-

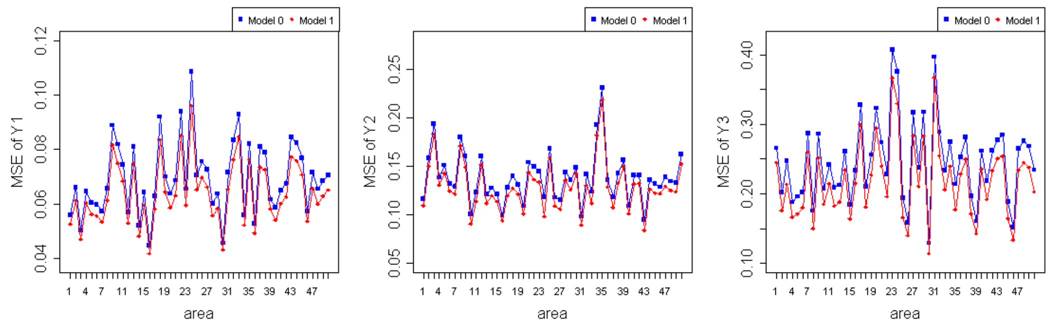


Figure 2: MSE of EBLUPs under Model 0 and Model 1.

covariance of direct estimation « $v1, v2, v3, v12, v13$, and $v23$ ». We compare the effectiveness of the univariate model (Model 0) and autoregressive multivariate Fay-Herriot Model (Model 2) by their MSE. We use `eblupMFH2()` to estimate parameters based on Model 2. Then, we plot the EBLUP and MSE of these methods to compare them.

```
data('datasae2')

# model specifications
Fo <- list(f1=Y1~X1+X2,
            f2=Y2~X1+X2,
            f3=Y3~X1+X2)
vardir <- c("v1", "v2", "v3", "v12", "v13", "v23")

# EBLUP based on Model 0 and Model 2
u <- eblupUFH(Fo, vardir, data=datasae2)  # Model 0
m2 <- eblupMFH2(Fo, vardir, data=datasae2)  # Model 2
```

The EBLUPs based on Model 0 (univariate model) and Model 1 (autoregressive multivariate model) are shown in Figure 3. As it can be seen, both methods show an almost similar result. However, EBLUPs based on Model 2 has a lower MSE than the EBLUPs based on Model 0, as shown in Figure 4. In this example, the autoregressive multivariate Fay-Herriot model (Model 2) seems to be more efficient than the univariate Fay-Herriot model (Model 0).

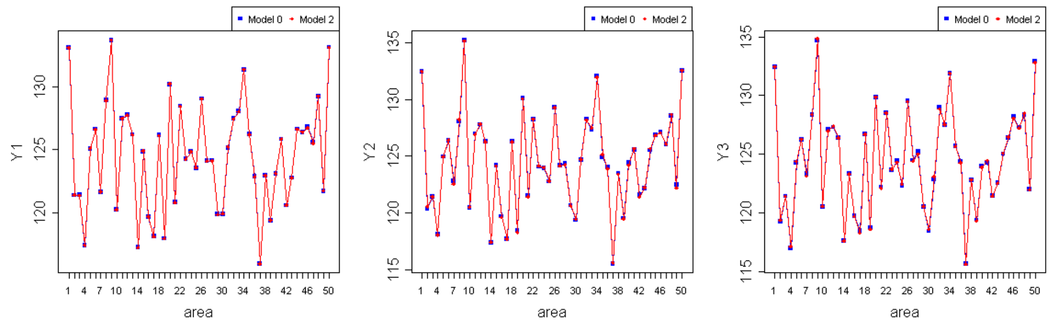


Figure 3: EBLUPs under Model 0 and Model 2.

Example 3. Heteroskedastic autoregressive multivariate Fay-Herriot model (Model 3)

`datasae3`, which was generated based on Model 3, is structured the same as `datasae1` and `datasae2`. We compare the effectiveness of the univariate model (Model 0) and heteroskedastic autoregressive multivariate Fay-Herriot Model (Model 3) by their MSE. We use `eblupMFH3()` to estimate parameters based on Model 3. Then, we plot the EBLUP and MSE of these methods to compare them.

```
data('datasae3')
```

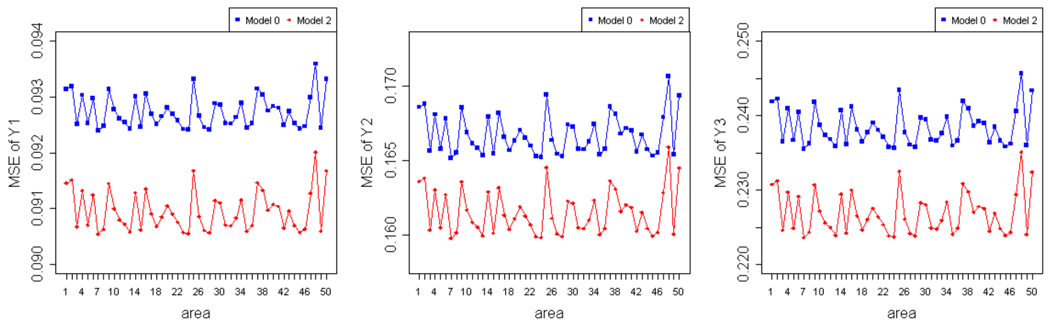


Figure 4: MSE of EBLUPs under Model 0 and Model 2.

```
# model specifications
Fo <- list(f1=Y1~X1+X2,
           f2=Y2~X1+X2,
           f3=Y3~X1+X2)
vardir <- c(”v1”, ”v2”, ”v3”, ”v12”, ”v13”, ”v23”)

# EBLUP based on Model 0 and Model 3
u <- eblupUFH(Fo, vardir, data=datasae3)  # Model 0
m3 <- eblupMFH3(Fo, vardir, data=datasae3)  # Model 3
```

Figure 5 shows EBLUPs based on Model 3 compared to Model 0. The MSEs of both methods are shown in Figure 6. It can be seen that the multivariate EBLUPs will follow the pattern of univariate EBLUPs with smaller MSE values. It can be concluded that the heteroskedastic autoregressive multivariate Fay-Herriot model (Model 3) is more efficient than the univariate model (Model 0).

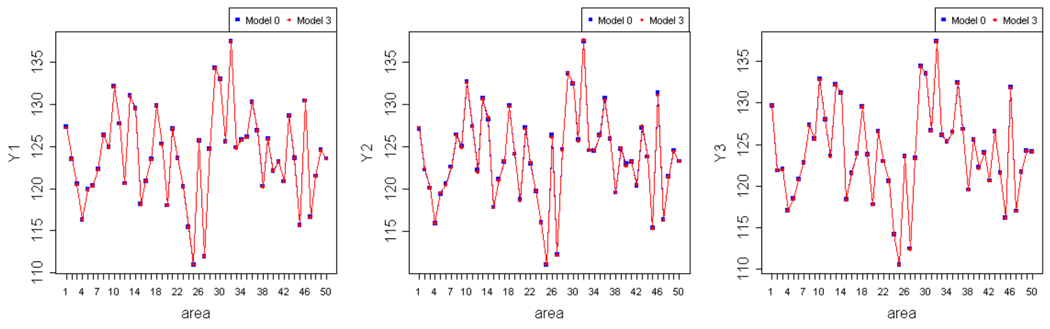


Figure 5: EBLUPs under Model 0 and Model 3.

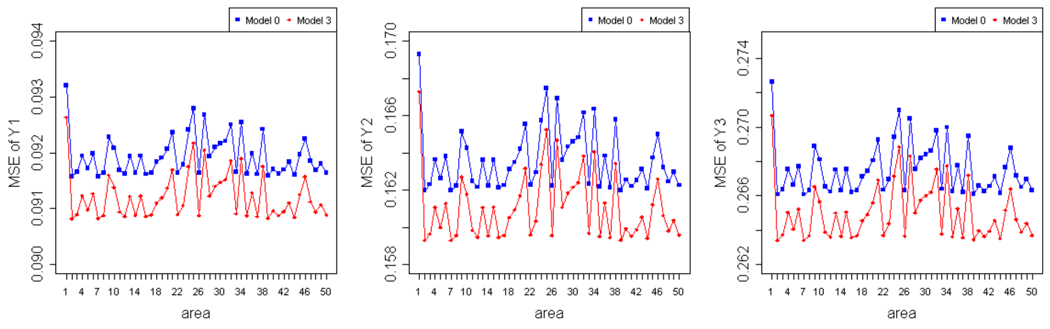


Figure 6: MSE of EBLUPs under Model 0 and Model 3.

Real data example: Poverty index

In this section, we use incomedata dataset, which is provided in library **sae**. The dataset contains unit-level data on income and other related variables in Spain. We will demonstrate how to estimate the EBLUP of Foster-Greer-Thorbecke (FGT) poverty index for each province using the multivariate Fay-Herriot Models. The index consists of three indicators, i.e., poverty proportion (p0), poverty gap (p1), and poverty severity (p2).

```
library(sae)
data("incomedata")
```

Firstly, we obtain area-level data by calculating poverty indicators for each unit and aggregating it by province based on Benavent and Morales (2016). We use poverty line z=6557.143 (Molina and Marhuenda, 2015). These following R commands are run to obtain $p0$, $p1$, and $p2$ as variables of interest.

```
library(tidyverse)

pov.line <- rep(6557.143, dim(incomedata)[1]) # poverty line

# calculate unit indictators
incomedata$y <- (pov.line - incomedata$income)/pov.line
incomedata = incomedata %>% mutate(poverty = ifelse(incomedata$y > 0, TRUE, FALSE),
                                   y0 = ifelse(incomedata$y > 0, incomedata$y^0, 0),
                                   y1 = ifelse(incomedata$y > 0, incomedata$y^1, 0),
                                   y2 = ifelse(incomedata$y > 0, incomedata$y^2, 0))

# estimated domain size
est.Nd <- aggregate(incomedata$weight, list(incomedata$prov), sum)[,2]

## estimate P0 P1 dan P2
prov.est = incomedata %>% group_by(prov) %>%
  summarise(p0.prov = sum(weight*y0),
            p1.prov = sum(weight*y1),
            p2.prov = sum(weight*y2)) %>%
  mutate(p0.prov = p0.prov / est.Nd,
         p1.prov = p1.prov / est.Nd,
         p2.prov = p2.prov / est.Nd)
incomedata <- incomedata %>% left_join(prov.est, by = c("prov" = "prov"))
```

We also need variance and covariance of variables of interest to estimate using the multivariate Fay-Herriot model. The following R commands are run to obtain variance and covariance of direct estimation based on Benavent and Morales (2016).

```
# estimate  direct estimation variance-covariance
prov.variance = incomedata %>%
  mutate(v11 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y0-p0.prov)*(y0-p0.prov), 0),
         v12 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y0-p0.prov)*(y1-p1.prov), 0),
         v13 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y0-p0.prov)*(y2-p2.prov), 0),
         v22 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y1-p1.prov)*(y1-p1.prov), 0),
         v23 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y1-p1.prov)*(y2-p2.prov), 0),
         v33 = ifelse(incomedata$poverty,
                      (weight*(weight-1))*(y2-p2.prov)*(y2-p2.prov), 0)) %>%
  group_by(prov) %>%
  summarise_at(c("v11","v12", "v13", 'v22',"v23","v33"), sum) %>%
  mutate_at(c("v11","v12", "v13", 'v22',"v23","v33"),
            function(x){x/est.Nd^2})
```

We use six explanatory variables selected by stepwise method, i.e., an indicator of age group 50-64 ($age4$), an indicator of age group >=65 ($age5$), an indicator of education level 1 ($educ1$), an indicator of education level 2 (*educ*2), an indicator of education level 3 (*educ*3), and an indicator of Spanish nationality (*nat*1). The model specifications are written as follow:

```
formula <- list(f1=p0.prov~age4+age5+educ1+educ2+educ3+nat1,
            f2=p1.prov~age4+age5+educ1+educ2+educ3+nat1,
            f3=p2.prov~age4+age5+educ1+educ2+educ3+nat1)
vardir <- c("v11", "v22", "v33", "v12", "v13", "v23")
```

Next, we select the most suitable multivariate Fay-Herriot model using variance homogeneity test and random effect ρ parameter test. In the variance homogeneity test, we test $H0 : \hat{\sigma}^2_{ui} = \hat{\sigma}^2_{uj}\, ; i, j = 1, 2, 3$ using model 3. We obtain a non-convergent model, the p-values are 0.98674 and 0.98996. It shows that the difference between variance of random effects is statistically not significant. After that, we test $H0 : \rho = 0$ using model 2. We get the t-statistics value of 19.26 with p-value of 0.00. It shows that there is a correlation between random effects. These results indicate the model that fits the data is Model 2.

The following codes are run to obtain the EBLUPs (under Model 2), to plot the direct estimates and the EBLUPs, and to plot the MSEs of direct estimates and the MSEs of EBLUPs ordered by sample size:

```
# EBLUP based on Model 2
eblup.pov <- eblupMFH2(formula, vardir, data=prov.data)

# Dataframe of Result
result_eblup = data.frame(prov = prov.data$prov,
                          est.Nd = est.Nd,
                          p0 = prov.data$p0.prov, p0.eblup = eblup.pov$eblup$p0.prov,
                          p1 = prov.data$p1.prov, p1.eblup = eblup.pov$eblup$p1.prov,
                          p2 = prov.data$p2.prov, p2.eblup = eblup.pov$eblup$p2.prov,
                          v11 = prov.data$v11, p0.mse = eblup.pov$MSE$p0.prov,
                          v22 = prov.data$v22, p1.mse = eblup.pov$MSE$p1.prov,
                          v33 = prov.data$v33, p2.mse = eblup.pov$MSE$p2.prov)
result_eblup = result_eblup %>% arrange(est.Nd)
result_eblup$prov = as.factor(result_eblup$prov)
result_eblup$id = 1:nrow(result_eblup)

# Figure 7: Direct estimates estimates and EBLUPs (under Model 2) ordered by sample size.
par(mfrow=c(1,3))

plot(result_eblup$id, result_eblup$p0,  type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p0", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p0.eblup, type = "o", col = "red", pch = 18)
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))
plot(result_eblup$id, result_eblup$p1, type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p1", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p1.eblup, type = "o", col = "red", pch = 18)
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))
plot(result_eblup$id, result_eblup$p2, type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p2", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p2.eblup, type = "o", col = "red", pch = 18)
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))

# Figure 8: MSE of Direct estimates and MSE of EBLUPs (under Model 2) ordered by sample size.
par(mfrow=c(1,3))

plot(result_eblup$id, result_eblup$v11, type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p0", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p0.mse, type = "o", col = "red", pch = 18)
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))
plot(result_eblup$id, result_eblup$v22, type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p1", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p1.mse, type = "o", col = "red", pch = 18)
```

```
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))
plot(result_eblup$id, result_eblup$v33, type = "o", col = "blue", pch = 15, xlab = "area",
    ylab = "p2", cex.axis = 1.5, cex.lab = 1.5, xaxt = "n")
points(result_eblup$p2.mse, type = "o", col = "red", pch = 18)
axis(1, at=result_eblup$id, labels = result_eblup$prov)
legend("topright",legend=c("Direct estimates","Model 2"),col=c("blue","red"), pch=c(15,18))
```

Variable of Interest	Statistic	Direct Estimation	Model 2
p0	Min	0.05244	0.04601
	Quartil 1	0.17296	0.18947
	Median	0.21850	0.21937
	Mean	0.22659	0.22020
	Quartil 3	0.27753	0.25468
	Max	0.43588	0.35011
	Standard Deviation	0.08173	0.056597
p1	Min	0.01891	0.02551
	Quartil 1	0.05336	0.05969
	Median	0.06810	0.06866
	Mean	0.07567	0.07407
	Quartil 3	0.09208	0.08866
	Max	0.15973	0.14023
	Standard Deviation	0.03227	0.02506
p2	Min	0.005449	0.008481
	Quartil 1	0.025819	0.026867
	Median	0.032344	0.033600
	Mean	0.039042	0.038179
	Quartil 3	0.051379	0.049755
	Max	0.099308	0.087194
	Standard Deviation	0.02083	0.017137

Table 1: Statistics of direct estimation and Model 2

We will compare EBLUPs based on Model 2 with the direct estimates. Both of the estimation results can be seen in Table 1. On the median value, the result of estimated poverty indicators are relatively similar, ranging from 0.218-0.219 for p0, 0.0681-0.0687 for p1, and 0.0323-0.0336 for p2. Model 2 has a lower range and smaller standard deviation than the direct estimation. It means that, in general, the multivariate Fay-Herriot model has lower variability of small area estimates than direct estimation.

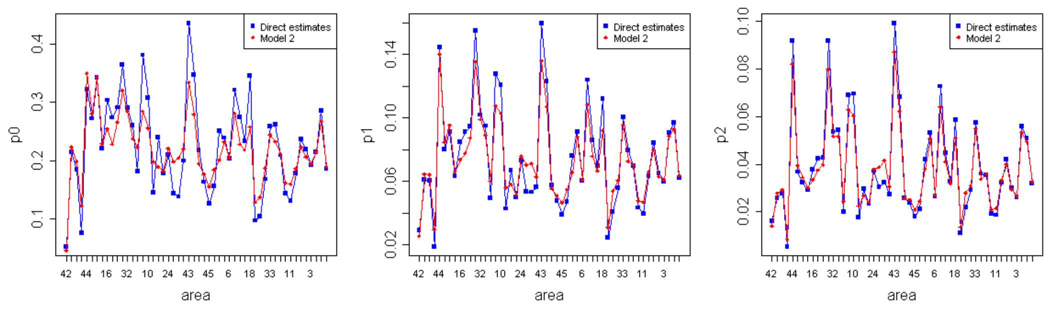


Figure 7: Direct estimates and EBLUPs (under Model 2) of p0, p1, and p2 ordered by sample size.

The results of direct estimates and EBLUPs under Model 2 ordered by sample size are shown in Figure 7. The patterns of small area estimates for both methods are almost the same for all areas. The MSEs of the direct estimates and the EBLUP estimates ordered by sample size are shown in Figure 8. These plots show that the multivariate Fay-Herriot model has lower MSE than the direct estimation. Thus, we can conclude that the multivariate Fay-Herriot model is more efficient than direct estimation.

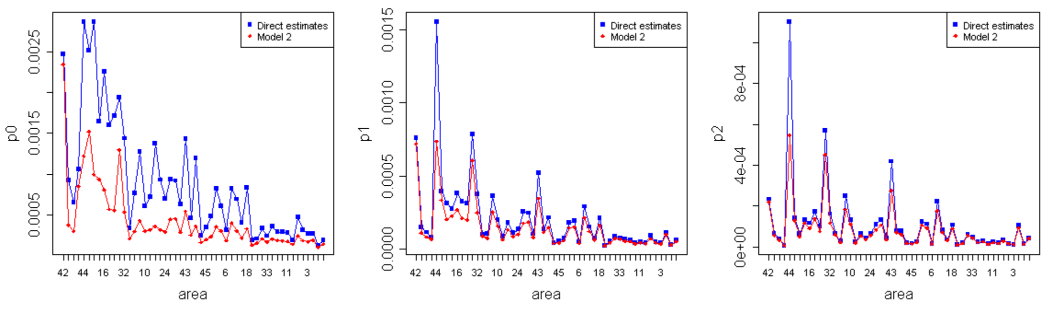


Figure 8: MSE of direct estimates and MSE of EBLUPs (under Model 2) of p0, p1, and p2 ordered by sample size.

Conclusion

This paper introduces the first R package of multivariate Fay-Herriot model for small area estimation named **msae**. The package is available on Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=msae`. This package contains a number of functions for estimating the EBLUP and MSE of EBLUP of each Fay-Herriot Model. This package accommodates the univariate Fay-Herriot model (model 0), multivariate Fay-Herriot model (model 1), autoregressive multivariate Fay-Herriot model (model 2), and heteroskedastic autoregressive multivariate Fay-Herriot model (model 3). The functions are described and implemented using three examples of datasets provided in **msae** package and a real dataset provided in **sae** package, `incomedata`. By these examples, we show that the multivariate Fay-herriot models produce more efficient parameter estimates than direct estimation and univariate model.

Bibliography

R. Benavent and D. Morales. Multivariate fay–herriot models for small area estimation. *Computational Statistics and Data Analysis*, page 372–390, 2016. URL `https://doi.org/10.1016/j.csda.2015.07.013`. [p111, 112, 113, 118]

H. J. Boonstra. *hbsae: Hierarchical Bayesian Small Area Estimation*, 2012. URL `https://CRAN.R-project.org/package=hbsae`. R package version 1.0. [p111]

J. Breidenbach. *JoSAE: Unit-Level and Area-Level Small Area Estimation*, 2018. URL `https://CRAN.R-project.org/package=JoSAE`. R package version 0.3.0. [p111]

Chengchun Shi Developer. *BayesSAE: Bayesian Analysis of Small Area Estimation*, 2018. URL `https://CRAN.R-project.org/package=BayesSAE`. R package version 1.0-2. [p111]

G. S. Datta, R. E. Fay, and M. Ghosh. Hierarchical and empirical multivariate bayes analysis in small area estimation. In *Proc. of the Bureau of the Census Annual Research Conference*, pages 63–79. Bureau of the Census, Washington D.C., 1991. [p111]

M. Ghosh and J. N. K. Rao. Small area estimation: An appraisal. *Statistical Science*, pages 55–93, 1994. [p111]

S. Krieg, H. J. Boonstra, and M. Smeets. Small area estimation with zero-inflated data - a simulation study. *Statistics Netherland*, pages 1–45, 2015. [p111]

M. D. E. Lefler, D. M. Gonzalez, and A. P. Martin. *saery: Small Area Estimation for Rao and Yu Model*, 2014. URL `https://CRAN.R-project.org/package=saery`. R package version 1.0. [p111]

E. Lopez-Vizcaino, M. Lombardia, and D. Morales. *mme: Multinomial Mixed Effects Models*, 2019. URL `https://CRAN.R-project.org/package=mme`. R package version 0.1-6. [p111]

I. Molina and Y. Marhuenda. sae: An r package for small area estimation. *The R Journal*, pages 81–98, 2015. URL `https://doi.org/10.32614/RJ-2015-007`. [p118]

I. Molina and Y. Marhuenda. *sae: Small Area Estimation*, 2018. URL `https://CRAN.R-project.org/package=sae`. R package version 1.2. [p111]

J. Pinheiro, D. Bates, and R-core. *nlme: Linear and Nonlinear Mixed Effects Models*, 2020. URL `https://CRAN.R-project.org/package=nlme`. R package version 3.1-145. [p111]

J. N. K. Rao and I. Molina. *Small Area Estimation 2nd Edition*. John Wiley and Sons Inc., Hoboken, New Jersey, 2015. [p111]

T. Schoch. *rsae: Robust Small Area Estimation*, 2014. URL `https://CRAN.R-project.org/package=rsae`. R package version 0.1-5. [p111]

M. Templ. *CRAN Task View: Official Statistics & Survey Methodology*, 2014. URL `https://CRAN.R-project.org/view=OfficialStatistics`. Version 2014-08-18. [p111]

A. Ubaidillah, K. A. Notodiputro, A. Kurnia, and I. W. Mangku. Multivariate fay-herriot models for small area estimation with application to household consumption per capita expenditure in indonesia. *Journal of Applied Statistics*, pages 2845–2861, 2019. URL `https://doi.org/10.1080/02664763.2019.1615420`. [p111]

Novia Permatasari
Politeknik Statistika STIS
East Jakarta
Indonesia
`16.9335@stis.ac.id`

Azka Ubaidillah
Politeknik Statistika STIS
East Jakarta
Indonesia
`azka@stis.ac.id`

cat.dt: An R package for fast construction of accurate Computerized Adaptive Tests using Decision Trees

by Javier Rodríguez-Cuadrado, Juan C. Laria and David Delgado-Gómez

Abstract This article introduces the **cat.dt** package for the creation of Computerized Adaptive Tests (CATs). Unlike existing packages, the **cat.dt** package represents the CAT in a Decision Tree (DT) structure. This allows building the test before its administration, ensuring that the creation time of the test is independent of the number of participants. Moreover, to accelerate the construction of the tree, the package controls its growth by joining nodes with similar estimations or distributions of the ability level and uses techniques such as message passing and pre-calculations. The constructed tree, as well as the estimation procedure, can be visualized using the graphical tools included in the package. An experiment designed to evaluate its performance shows that the **cat.dt** package drastically reduces computational time in the creation of CATs without compromising accuracy.

Introduction

Nowadays, there is an increasing interest in the development and application of Computerized Adaptive Tests (CATs). For instance, they are applied in several areas such as psychology (Ma et al., 2017; Mizumoto et al., 2019), education (He and Min, 2017; Wu et al., 2017), or medicine (Michel et al., 2018; Fox et al., 2019). The reason behind their popularity is that CATs can estimate the ability level of a psychological variable of interest in an examinee with greater accuracy than the classical tests by administering a smaller number of items (Weiss, 2004). Besides, the existence of certain mechanisms, such as item exposure control (Georgiadou et al., 2007) limits the leaking of items between participants.

Concisely, CATs are tailored tests. Every item administered to the examinee is chosen from an item bank by employing a selection criterion that considers: i) the answers given by the participant to the items previously administered; ii) the characteristics of such items, and iii) the probabilities provided by a model that relates the responses to each item with its characteristics. The most commonly used criterion is Maximum Fisher Information (MFI) (Zhou and Reckase, 2014; Li et al., 2020), which selects the item that provides the highest information for the current estimate of the ability level. However, this criterion presents several drawbacks. These include item selection bias, large estimation errors at the beginning of the test, high item exposure rates, and content imbalance problems (Ueno, 2013; ZhuoKang and Liu, 2012). Various alternatives have been proposed as attempts for addressing these problems. Among them stand out Minimum Expected Posterior Variance (MEPV) (Van der Linden and Pashley, 2009), Kullback-Leibler Information (Chang and Ying, 1996), and Maximum Likelihood Weighted Information (Veerkamp and Berger, 1997). Although these selection techniques largely solve the aforementioned problems, their high computational cost complicates their practical use.

Decision Trees (DTs) have been proposed to reduce the computational cost in the creation of CATs. Yan et al. (2004) used regression trees to predict the participants' total score. A remarkable feature of this work is the merge of nodes to maintain a sufficient sample size to perform the partitions. Afterward, Ueno and Songmuang (2010) developed an item selection criterion based on mutual information in regression trees. However, unlike CATs based on Item Response Theory (IRT), these works predict the total score of the participant rather than estimate their ability level, which makes it difficult to compare the performance of a participant in two different tests aimed at measuring the same construct. Recently, Delgado-Gómez et al. (2019) mathematically proved the equivalence between IRT-based CATs and DTs when the MEPV item selection criterion is used, proposing the Tree-CAT method, which integrates both methodologies. In their method, each node of the tree contains an item, emerging from it as many branches as the number of the item's possible responses. The examinees progress through the tree according to the responses they provide until reaching the last node, where their final estimate corresponds to the found ability level. The disadvantage of this method is that it requires a high-performance cluster to create the tree. In this regard, the Merged Tree-CAT method (Rodríguez-Cuadrado et al., 2020) extends and improves Tree-CAT, accelerating tree construction by joining nodes with similar estimates or ability level distributions.

Currently, there are several packages oriented to the creation of CATs in R. Among them, we can find **catR** (Magis et al., 2012; Magis and Barrada, 2017), **mirtCAT** (Chalmers, 2016), and **catIrt** (Nydick, 2014). The drawback of these packages is that they create a separate CAT for each examinee, reducing their efficiency. For example, if two individuals provided the same response to the first item, the computations for estimation and selection of the next item would be performed twice, even though the result would be the same. Therefore, the shortage of memory in these packages considerably increases computational time, which makes the practical application of CATs difficult, or even impossible, when the item selection criterion is one of the most computationally demanding.

This article describes the **cat.dt** package in which the Merged Tree-CAT method (Rodríguez-Cuadrado et al., 2020) is implemented. Unlike the existing packages, **cat.dt** creates the CAT before it is administered to the examinees and stores it in a DT structure. This allows that each time the participant responds to an item, the estimation of their ability level and the selection of the next item to be administered is immediate since it is pre-computed. This differs from the existing packages, in which both calculations are performed during the test administration, making it difficult or even impossible to use computationally expensive item selection criteria such as the MEPV. In this way, the **cat.dt** package manages to quickly create CATs on a standard personal computer as well as provides accurate estimates of the ability level of each examinee.

The rest of the article is structured as follows. Firstly, it introduces the elements of IRT and CATs used by the Merged Tree-CAT. Next, the functions contained in the package are detailed, and an example of its use is provided. Then, the performance of the **cat.dt** package is compared to that of the **catR** package. Finally, the article discusses the benefits of the package.

IRT and CATs

The IRT, on which CATs are based, assumes that a participant's response to an item depends on the ability level of the individual and the characteristics of that item (Richardson, 1936; Lawley, 1943; Tucker, 1946). This relationship is obtained through probabilistic models in which the probability $P_{ik}(\theta, \pi_i)$ that an examinee gives the response $k = 1, \ldots, K_i$ to an item i depends on their ability level θ and the parameters of the item π_i.

For polytomous items, the most widespread model for ordered responses is the Graded Response Model (GRM) developed by Samejima (1969):

$$P^*_{ik}(\theta, \pi_i) = \frac{e^{\alpha_i(\theta-\beta_{ik})}}{1+e^{\alpha_i(\theta-\beta_{ik})}}, \tag{1}$$

where α_i is the discrimination parameter, β_{ik} are the difficulty (or location) parameters of each response k, and $P^*_{ik}(\theta, \pi_i)$ the probability of giving the response k or greater. Therefore $P_{ik}(\theta, \pi_i) = P^*_{ik}(\theta, \pi_i) - P^*_{ik+1}(\theta, \pi_i)$, being $P^*_{i1}(\theta, \pi_i) = 1, \ldots, P^*_{iK_i+1}(\theta, \pi_i) = 0$.

When there is not a particular order in the responses, the most generic model is the Nominal Response Model (NRM), defined by Bock (1972):

$$P_{ik}(\theta, \pi_i) = \frac{e^{\rho_{ik}\theta+\gamma_{ik}}}{\sum_{r=1}^{K_i} e^{\rho_{ir}\theta+\gamma_{ir}}}, \tag{2}$$

being ρ_{ik} and γ_{ik} the slope and intercept parameters, respectively, for item i and response k.

These probabilistic models are used in the CATs to obtain the estimate of the ability level of the examinee based on their responses. Of all the existing estimation methods, the Expected a Posteriori (EAP) technique is widely used given the simplicity of its calculation and the minimum Mean Squared Error (MSE) of its estimations (Bock and Mislevy, 1982). When the responses $R_{i_1}, \ldots, R_{i_M}$ of the examinee to the items $i_i, \ldots, i_M$ are the possible responses $k_1, \ldots, k_M$ of those items, the estimate $\hat{\theta}$ of the ability level is given by:

$$\hat{\theta} = \int_{-\infty}^{\infty} \theta f(\theta \mid R_{i_1} = k_1, \ldots, R_{i_M} = k_M) d\theta, \tag{3}$$

being $f(\theta \mid R_{i_1} = k_1, \ldots, R_{i_M} = k_M)$ the posterior density function given the responses according to Bayes' theorem:

$$f(\theta \mid R_{i_1} = k_1, \ldots, R_{i_M} = k_M) = \frac{P_{i_1k_1}(\theta, \pi_{i_1}) \cdots P_{i_Mk_M}(\theta, \pi_{i_M}) f(\theta)}{\int_{-\infty}^{\infty} P_{i_1k_1}(\theta, \pi_{i_1}) \cdots P_{i_Mk_M}(\theta, \pi_{i_M}) f(\theta) d\theta}, \tag{4}$$

where $f(\theta)$ is the prior density function of the examinee's ability level.

Each time an examinee responds to an item, this ability level estimation is used by the CAT to choose the next item. Among the existing item selection methods, MFI is the most popular. This criterion consists of choosing the item i^* that maximizes the Fisher information function $F_i(\theta)$ evaluated at the current estimate $\hat{\theta}$ of the ability level. This function is given by (Magis, 2015):

$$F_i(\theta) = \sum_{k=1}^{K_i} \frac{P'_{ik}(\theta,\pi_i)^2}{P_{ik}(\theta,\pi_i)} - P''_{ik}(\theta,\pi_i) \tag{5}$$

Another criterion to highlight is the MEPV because it has been shown to be equivalent to minimizing the MSE of the estimates of the ability level (Delgado-Gómez et al., 2019). In this case, given the posterior density function $f(\theta \mid X_{i_1} = k_1, \ldots, X_{i_M} = k_M)$ obtained from the responses to the items $i_1, \ldots, i_M$, the MEPV criterion chooses the item i^* that minimizes the function:

$$E_i = \int_{-\infty}^{\infty} \left(\sum_{k=1}^{K_i} \left(\theta - \hat{\theta}^i_{M+1,k}\right)^2 P_{ik}(\theta,\pi_i) \right) f(\theta \mid R_{i_1} = k_1, \ldots, R_{i_M} = k_M) d\theta, \tag{6}$$

being $\hat{\theta}^i_{M+1,k}$ the ability level estimation if the examinee's response R_{M+1} is the possible response k of item i.

In summary, CATs are constructed as follows. Starting from a prior density $f(\theta)$ and a prior estimate θ_0, the first item to be administered to the examinee is selected according to an established criterion. Once a response is given, their ability level is estimated, from which the next item to administer is selected. This process is repeated until a stopping criterion is reached (for example, a predetermined number of items to be administered per participant), being the final estimation of the examinee the one obtained after their last response.

The aforementioned process is structured by the Merged Tree-CAT method in a DT as follows (Rodríguez-Cuadrado et al., 2020). As in the Tree-CAT method, each tree node has an assigned item and an associated estimate based on the responses given to the items assigned to the parent nodes. The novelty of the Merged Tree-CAT method consists of limiting the growth of the tree by joining nodes with similar estimates, accelerating its construction with the least loss of precision in the estimates. Once constructed, the examinee progresses through the tree according to their answers until reaching the last node, where the final estimate of their ability level is found. In addition, the Merged Tree-CAT method incorporates item exposure control by establishing an exposure rate that limits the percentage of participants that are administered with each item, which increases the safety of the test.

The **cat.dt** package implements the Merged Tree-CAT method. As it will be explained in the following section, each of the items that form the test is chosen using the MFI or MEPV criterion, employing the estimate obtained by the EAP method according to the GRM or NRM model.

The cat.dt package

This section starts by describing the **cat.dt** package architecture and its main function `CAT_DT`. This is followed by a practical example on how to use this package to create CATs structured in DTs, to visualize them, and to obtain estimates of the ability level of participants. Finally, we detail some of the computational features taken into account in the building of the package to increase its efficiency.

The **cat.dt** package can be installed from CRAN (`install.packages("cat.dt")`) or from the development version's GitHub repository `https://github.com/jlaria/cat.dt`. [1]

cat.dt structure

The **cat.dt** package consists of the functions shown in Figure 1, which also displays the dependency relationships between them.

The most relevant functions are the following:

- `CAT_DT`: Creates the CAT structured in a DT.
- `create_level_1`: Creates the nodes that conform to the first level of the DT.
- `create_levels`: Creates the nodes that conform to the levels of the DT (except for the first level).
- `join_node`: Joins nodes from the same tree level with similar estimations or distributions of the ability level.
- `CAT_ability_est` : Estimates the ability level of a participant after each response and computes a Bayesian credible interval of the final estimation.

[1]This package imports the dependencies **Matrix**, **Rglpk**, and **ggplot2** for matrix treatment, linear programming, and visualization, respectively.

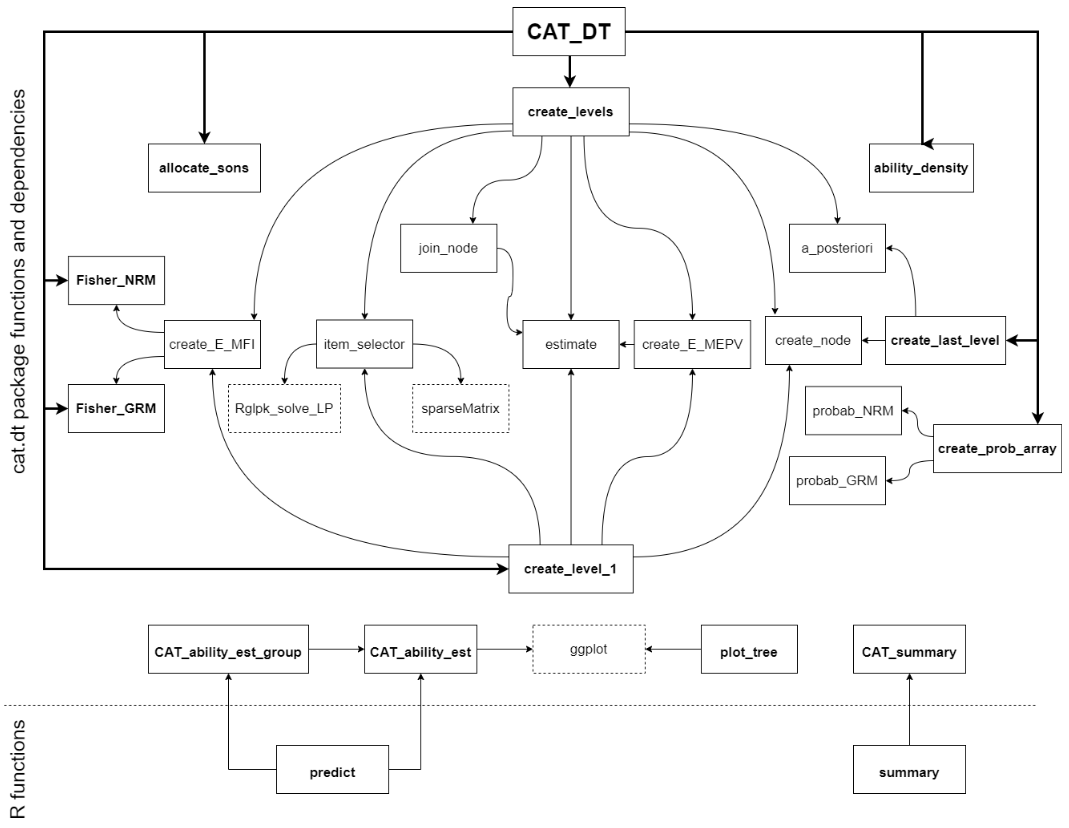


Figure 1: Function dependencies.

The `CAT_DT` function

The input parameters of the main function `CAT_DT` are introduced in Table 1.

The tree growth is controlled by the parameters `limit`, `inters`, and `p`. The `limit` parameter is the maximum number N of nodes per level. When this number is exceeded in the construction of the tree, those nodes whose estimates of the ability level are at a distance less than a threshold $(\lambda_L - \lambda_U)/N$ are joined, being λ_L and λ_U the lower and upper bounds of an interval with probability `p` according to the prior density function of the ability level. Finally, `inters` is the minimum value that must exceed the intersection between the density functions of two nodes to join when the maximum number N of nodes per level has not been reached. This intersection is obtained using the methodology defined in Cha (2007).

Finally, the `CAT_DT` function returns a list with the input parameters introduced by the user and the elements described in Table 2.

cat.dt usage example

Firstly, it is shown how to build a CAT using the main function `CAT_DT`. To do this, the item bank from the data frame `itemBank` included in the package is used. Given the nature of these items, the probabilistic model used is the GRM. Also, the item selection criterion adopted is the MEPV, the exposure rate is set at 0.3, the length of the test at 10 items, and the prior distribution of the ability level at an $N(0, 1)$, leaving the rest of the parameters at their default values. The function call is made as follows:

```
example_cat <- CAT_DT(bank = itemBank, model = "GRM", crit = "MEPV", C = 0.3,
stop = c(10,0), limit = 200, inters = 0.98, p = 0.9, dens = dnorm, 0, 1)
```

Among the values returned by this function, the list `nodes` contains all the nodes that conform the DT in which the CAT is structured. These nodes are grouped by levels. As an example, if we access the first node of the third level,

```
example_cat$nodes[[3]][[1]]
```

we obtain

`bank`	Item bank. It must be a data frame in which each row represents an item and each column one of its parameters. If the model is GRM, the first column must be the discrimination parameter and the remaining columns are the difficulty (or location) parameters (Samejima, 1969). If the model is NRM, the odd columns must be the slope parameters and the even columns the intercept parameters. (Bock, 1972).
`model`	CAT probabilistic model. Options: "`GRM`" (default) and "`NRM`".
`crit`	Item selection criterion. Options: "`MEPV`" for the Minimum Expected Posterior Variance (default) or "`MFI`" for the Maximum Fisher Information.
`C`	Expected fraction C of participants administered with each item (exposure rate). It can be a vector with as many elements as items in the bank or a positive number if all the items have the same rate. Default: `C = 0.3`.
`stop`	Vector of two components that represent the CAT stopping criteria. The first component represents the maximum level L of the DT and the second represents the threshold for the Standard Error (SE) of the ability level (Bock and Mislevy, 1982) (if 0, this second criterion is not applied). Default: `stop = c(6,0)`.
`limit`	Maximum number N of nodes per level (max. $N = 10000$). This is the main parameter that controls the tree growth. It must be a natural number. Default: `limit = 200`.
`inters`	Minimum intersection of the density functions of two nodes to be joined. It must be a number between 0 and 1. If the user wants to avoid using this criteria, `inters = 0` should be specified. Default: `inters = 0.98`.
`p`	Prior probability of the interval whose limits determine a threshold for the distance between estimations of nodes to join. Default: `p = 0.9`.
`dens`	Prior density function of the ability level. It must be an R function: `dnorm`, `dunif`, etc.
`...`	Parameters to `dens`.

Table 1: Main function parameters.

```
$`ID`
[1] 30001

$item
[1] 22

$item_prev
[1] 18 11

$est
[1] -1.10257

$SE
[1] 0.8071584

$ID_sons
```

`nodes`	List with a maximum of $L+1$ elements (levels). Each level contains a list of the nodes of the corresponding level. Note that the first level will contain more than one root node if $C < 1$. In this case, each examinee would start the test for one of them at random. The nodes of the additional level $L+1$ only include the estimation and distribution of the ability level given the responses to the items of the final level L. Note that the `nodes` list can have less than $L+1$ elements if the SE stopping criterion is satisfied for all the nodes from a previous level.
`C_left`	Residual exposure rate of each item after the CAT construction.
`predict`	Function that returns the estimated ability level of an examinee after each response and a Bayesian credible interval of the final estimation given their responses to the items from the item bank. These responses must be entered by the user as a numeric vector input. In addition, it returns a vector with the items that have been administered to the examinee. This is the function `CAT_ability_est` of the package.
`predict_group`	Function that returns a list whose elements are the returned values of the function `predict` for every examinee from the group. This is the function `CAT_ability_est_group` of the package.

Table 2: Main function output.

```
  ID_son Response Probability
1  40001        1           1
2  40002        2           1
3  40003        3           1

$D
[1] 0.03331851

$as_val
[1] 0.5396903
```

This list contains, among others, information about: i) `ID`, the node identifier; ii) `item`, the item assigned to the node; iii) `item_prev`, the items previously administered to the examinee that reaches the node; iv) `est`, the estimation of the ability level after their responses to these items, v) `SE`, the SE associated to that estimation. Finally, the data frame `ID_sons` contains the ID of each child node, the response that leads to it, and the probability of, given that response, accessing that child node.

A description of the tree can be obtained using the R function `summary`. This function provides: i) The number of levels; ii) The number of nodes per level; iii) The probabilistic model used; iv) The item selection criterion used; v) The residual exposure rate of every item and vi) The percentage of items used to build the test. In order to summarize the tree, we enter

```
summary(example_cat)
```

to obtain

```
----------------------------------------------------------------------
Number of tree levels: 10

Number of nodes in level 1 : 4
Number of nodes in level 2 : 14
Number of nodes in level 3 : 39
Number of nodes in level 4 : 99
Number of nodes in level 5 : 101
Number of nodes in level 6 : 124
Number of nodes in level 7 : 141
Number of nodes in level 8 : 158
Number of nodes in level 9 : 165
Number of nodes in level 10 : 177
```

```
----------------------------------------------------------------------
Psychometric probabilistic model: GRM
Item selection criterion: MEPV
----------------------------------------------------------------------
Item exposure:
item  1 : 0.000   item  2 : 0.0768   item  3 : 0.000   item  4 : 0.1704
item  5 : 0.1532   item  6 : 0.000   item  7 : 0.300   item  8 : 0.000
item  9 : 0.000   item  10 : 0.000   item  11 : 0.300   item  12 : 0.000
item  13 : 0.000   item  14 : 0.000   item  15 : 0.2508   item  16 : 0.000
item  17 : 0.000   item  18 : 0.300   item  19 : 0.2113   item  20 : 0.255
item  21 : 0.2314   item  22 : 0.2565   item  23 : 0.000   item  24 : 0.000
item  25 : 0.0532   item  26 : 0.000   item  27 : 0.000   item  28 : 0.000
item  29 : 0.000   item  30 : 0.0855   item  31 : 0.300   item  32 : 0.000
item  33 : 0.0107   item  34 : 0.000   item  35 : 0.000   item  36 : 0.0127
item  37 : 0.000   item  38 : 0.000   item  39 : 0.2467   item  40 : 0.0902
item  41 : 0.000   item  42 : 0.000   item  43 : 0.2611   item  44 : 0.300
item  45 : 0.000   item  46 : 0.000   item  47 : 0.300   item  48 : 0.0885
item  49 : 0.000   item  50 : 0.300   item  51 : 0.1192   item  52 : 0.000
item  53 : 0.0166   item  54 : 0.000   item  55 : 0.300   item  56 : 0.300
item  57 : 0.000   item  58 : 0.000   item  59 : 0.2233   item  60 : 0.000
item  61 : 0.300   item  62 : 0.000   item  63 : 0.2583   item  64 : 0.0839
item  65 : 0.000   item  66 : 0.0367   item  67 : 0.000   item  68 : 0.300
item  69 : 0.300   item  70 : 0.300   item  71 : 0.1213   item  72 : 0.000
item  73 : 0.1803   item  74 : 0.000   item  75 : 0.000   item  76 : 0.000
item  77 : 0.000   item  78 : 0.000   item  79 : 0.0153   item  80 : 0.000
item  81 : 0.000   item  82 : 0.000   item  83 : 0.300   item  84 : 0.000
item  85 : 0.2125   item  86 : 0.1251   item  87 : 0.1741   item  88 : 0.000
item  89 : 0.0253   item  90 : 0.300   item  91 : 0.000   item  92 : 0.300
item  93 : 0.2539   item  94 : 0.300   item  95 : 0.300   item  96 : 0.300
item  97 : 0.000   item  98 : 0.000   item  99 : 0.000   item  100 : 0.000

Percentage of items used: 49 %
----------------------------------------------------------------------
```

In addition, the tree created can be visualized by means of the function `plot_tree`. This function takes as input arguments: i) The tree created; ii) The number of levels to plot, iii) The index of the root node to start the test. For example, by introducing

```
plot_tree(example_cat, levels = 3, tree = 3)
```

we obtain the following plot

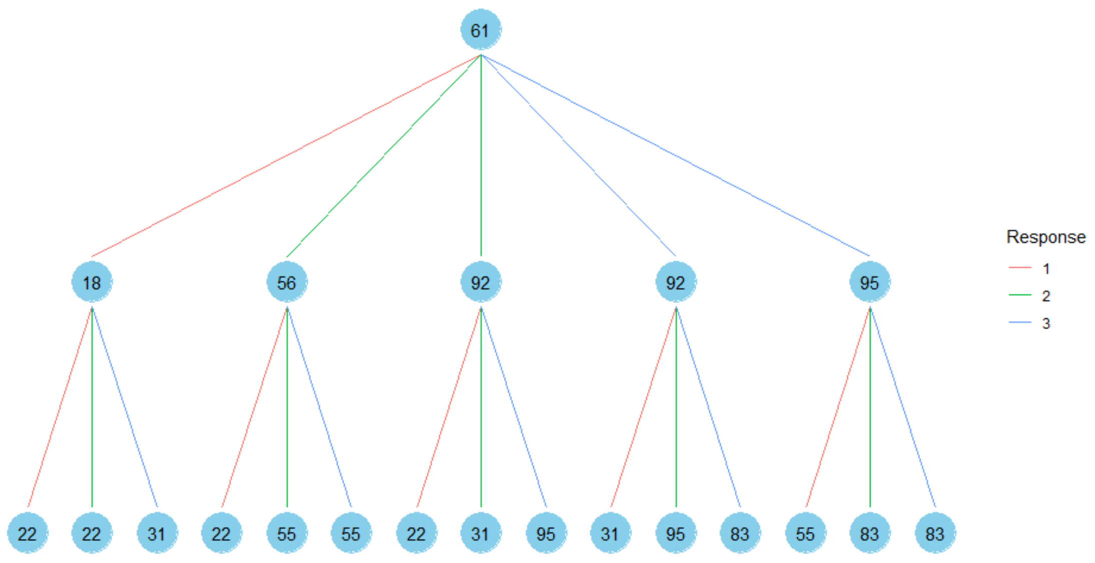


Figure 2: Tree visualization.

Once the CAT has been created, the ability level of an examinee is estimated using the `predict` function. The **cat.dt** package includes as an example the matrix `itemRes`, which contains the responses of 30 examinees to the 100 items of the bank `itemBank`. For instance, to calculate the estimate of the fifth examinee (a seed is needed to obtain the same result due to probabilistic node access) we use

```
set.seed(0)
predict(example_cat, itemRes[5, ])
```

We obtain

```
$`estimation`
 [1] 0.7649071 0.3114948 0.4320018 0.6780359 0.4762673 0.7993720 0.7504451 0.5928716
 [9] 0.7041479 0.6217778

$llow
[1] -0.325

$lupp
[1] 1.565

$items
[1] 70 83 95 55  4 39  7 51 96 73

$graphics
```

This output list contains: i) The vector `estimation`, which includes the estimated ability level after each response; ii) The lower `llow` and upper `lupp` bounds of a Bayesian credible interval at 95% of the final estimation of the examinee's ability level; iii) The vector `items`, which contains the items that have been administered to the examinee in the CAT, iv) The object `graphics`, which represents the evolution of the ability level estimation automatically as shown in 3.

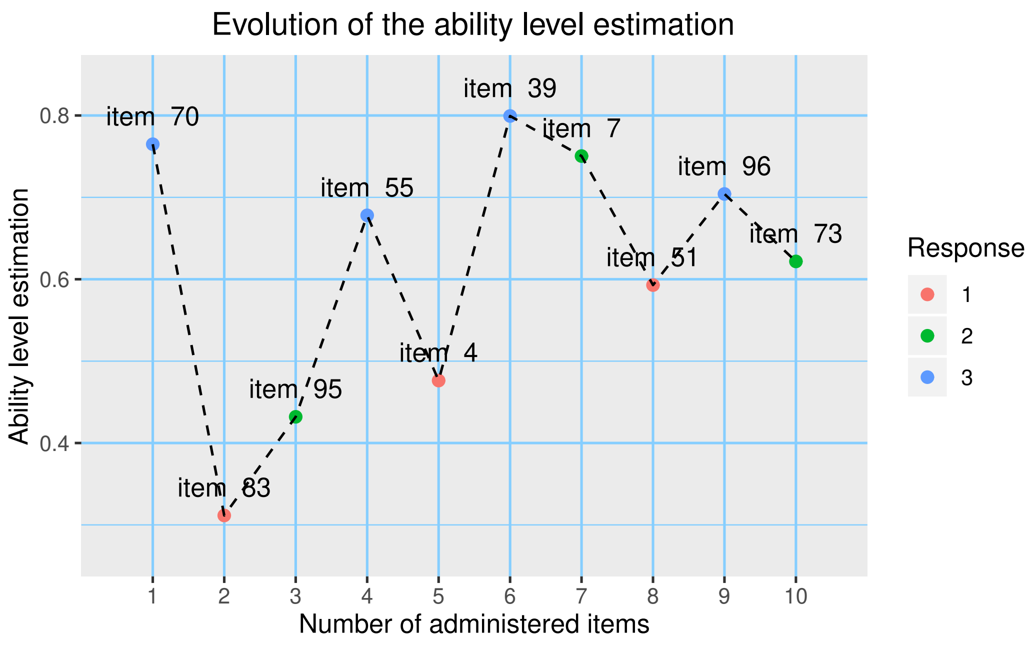


Figure 3: Evolution of the ability level estimation.

This plot represents the estimation of the ability level after responding to each administered item. For example, giving response 3 to item 70 results in an estimate of approximately $\hat{\theta} = 0.76$. Then, after giving response 1 to item 83, the estimate decreases to approximately $\hat{\theta} = 0.31$, and so on. Note that the value of the response influences whether the estimate decreases or increases. Alternatively to the `predict` function, it can be entered `CAT_ability_est(example_cat,itemRes[5,])` or `example_cat$predict(itemRes[5,])`.

Finally, these results can be obtained for a whole group of examinees also by calling the function `predict`. In this case, this function returns a list whose elements are the outputs as if it was called for every examinee. Once it is stored in a variable in the following way,

```
est_group <- predict(example_cat, itemRes)
```

the estimation information is available for every examinee. For instance, to know the items administered to the second examinee, the following must be introduced:

```
est_group[[2]]$items
```

Obtaining:

```
[1] 61 92 95 55 50 59 85 63 19 69
```

Alternatively to the `predict` function, it can be introduced `example_cat$predict_group(itemRes)` or `CAT_ability_est_group(example_cat,itemRes)`.

A similar example can be found in the tutorial vignette included in the package.

Computational features

Before ending this section, we detail the computational features that accelerate the construction of the tree and reduce memory space.

- Message passing: The calculation of the posterior density function (equation 4) is necessary to obtain the estimation of the ability level (equation 3), the selection of the next item according to the MEPV criterion (equation 6), and the Bayesian credible interval of the final estimation. Such calculation would involve the multiplication of M terms for each node of the corresponding level. However, since the multiplication of the first $M-1$ terms is done to obtain the density function of the parent node, this information is stored by the **cat.dt** package and passed to the child node, in which only the last term is multiplied.
- Joining nodes: the **cat.dt** package joins those nodes whose estimations and/or posterior density functions meet the similarity criteria determined by the parameters `limit`, `inters`, and `p`. These unions control tree growth, which significantly accelerates tree creation and reduces the amount of memory space without losing precision in the estimations.
- Riemann integration and probability pre-calculation: The integrals required in the equations 3, 4, and 6 are approximated numerically by Riemann integration. To do this, a set of ability levels $\{\theta_0, \ldots, \theta_{4000}\}$ is considered, where $\theta_j = -10 + j/200$, covering the interval $(-10, 10)$. The Riemann integration in equation 6 requires the previous calculation of $P_{ik}(\theta_j, \pi_i)$ for each item i, possible answer k, and ability level θ_j. Because of this, these probabilities are calculated and stored before the creation of the tree and then used in equations 4, 5, and 6. This avoids repeating unnecessary calculations and accelerates the creation of the CAT.

Performance assessment

In this section, the performance of the **cat.dt** package is compared to that of the **catR** package. Ability level estimates and computational times of both packages have been studied for nine different simulation scenarios. In each scenario, a CAT is constructed from an item bank (composed of 100, 200, or 500 items), which is administered to a group of examinees (1000, 2000, or 5000 examinees). The database for each scenario can be found online [2] in the format `[number of items]_items_[number of examinees]_examinees.RData`.

Similar to the example of the previous section, the probabilistic model used is GRM, the criteria for selecting items is MEPV, the length of the test is ten items per participant (the SE threshold is set at 0), and the prior distribution of the ability level is $N(0,1)$. However, unlike the aforementioned example, there is no item exposure control ($C = 1$) since the implementation differs in both packages: the **cat.dt** package builds the CAT before administration and the **catR** package during administration.

All the simulations for both packages were run in an HP Z230 Tower Workstation with an Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, with 32 GB of RAM, running Debian GNU/Linux 10, R 3.5.2.

Figure 4 illustrates the MSE of the ability level estimates after each response of the examinees, obtained by both packages in each scenario. It is observed that the estimates after each response are equally accurate, indicating that the performance of both packages is very similar.

On the other hand, Table 3 displays the time employed by each package in the computations for the creation of the CAT and the evaluation of those examined in each scenario. It can be seen that the **cat.dt** package is barely affected by the number of examinees since the CAT is created before it is administered to the participants. However, the **catR** package creates a CAT for each examinee, so the computational time is proportional to the number of participants. This causes **catR** package to take several days in total for the creation and administration of the CAT, whereas package **cat.dt** takes a few minutes, being this difference larger the higher the number of examinees.

[2] `https://github.com/jlaria/cat.dt-performance-assessment`

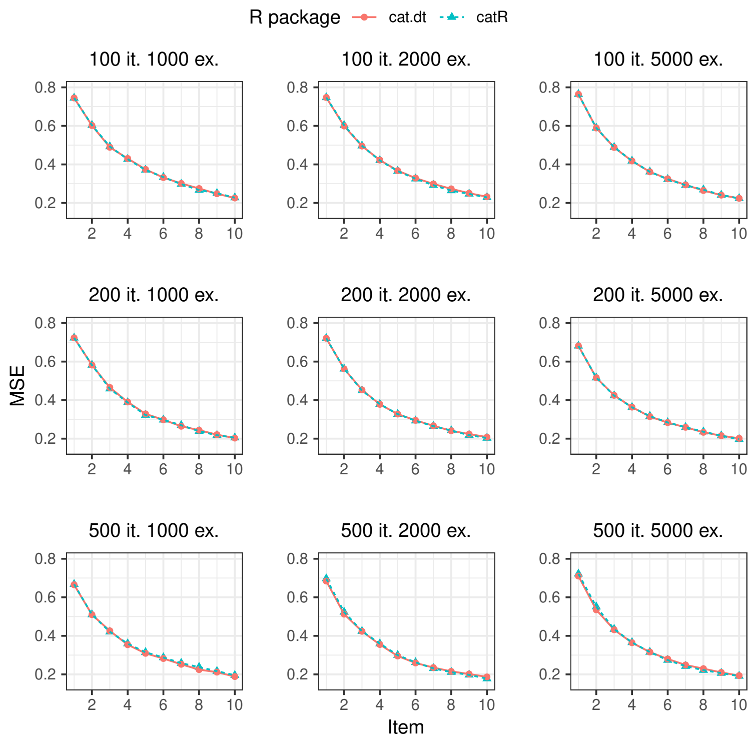


Figure 4: MSE of ability trait estimates.

		100 items		200 items		500 items	
		cat.dt	catR	cat.dt	catR	cat.dt	catR
exam.	1000	93.94 s	40.23 h	263.04 s	81.04 h	608.71 s	206.61 h
	2000	160.67 s	78.55 h	314.88 s	162.22 h	635.54 s	413.82 h
	5000	160.14 s	196.98 h	264.50 s	406.79 h	675.42 s	1033.65 h

Table 3: CAT creation and evaluation times.

Summary

This article has introduced the **cat.dt** package oriented to the creation of CATs structured in DTs, their visualization, and the estimation of the ability levels of the examinees. Unlike the existing packages, the **cat.dt** package creates the test before being administered to the examinees, so its performance is independent of the number of participants. For this reason, it is ideal for application to large groups, taking a few minutes to create and administer the test. Besides, it has been shown that the **cat.dt** package obtains ability level estimates as accurate as those obtained by the **catR** package, which is widely used in the field of CATs.

Acknowledgments

This article has been funded by the Spanish National Project No. RTI2018-101857-B-I00.

Bibliography

R. D. Bock. Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, 37(1):29–51, 1972. URL `https://doi.org/10.1007/BF02291411`. [p124, 127]

R. D. Bock and R. J. Mislevy. Adaptive eap estimation of ability in a microcomputer environment. *Applied psychological measurement*, 6(4):431–444, 1982. URL https://doi.org/10.1177/014662168200600405. [p124, 127]

S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical models and Methods in Applied Sciences*, 1(4):300–307, 2007. [p126]

R. P. Chalmers. Generating adaptive and non-adaptive test interfaces for multidimensional item response theory applications. *Journal of Statistical Software*, 71(5):1–39, 2016. URL https://doi.org/10.18637/jss.v071.i05. [p123]

H.-H. Chang and Z. Ying. A global information approach to computerized adaptive testing. *Applied Psychological Measurement*, 20(3):213–229, 1996. URL https://doi.org/10.1177/014662169602000303. [p123]

D. Delgado-Gómez, J. C. Laria, and D. Ruiz-Hernández. Computerized adaptive test and decision trees: A unifying approach. *Expert Systems with Applications*, 117:358–366, 2019. URL https://doi.org/10.1016/j.eswa.2018.09.052. [p123, 125]

R. S. Fox, P. I. Moreno, B. Yanez, R. Estabrook, J. Thomas, L. C. Bouchard, H. L. McGinty, D. C. Mohr, M. J. Begale, S. C. Flury, et al. Integrating promis® computerized adaptive tests into a web-based intervention for prostate cancer. *Health Psychology*, 38(5):403–409, 2019. URL http://dx.doi.org/10.1037/hea0000672. [p123]

E. G. Georgiadou, E. Triantafillou, and A. A. Economides. A review of item exposure control strategies for computerized adaptive testing developed from 1983 to 2005. *The Journal of Technology, Learning and Assessment*, 5(8), 2007. [p123]

L. He and S. Min. Development and validation of a computer adaptive efl test. *Language Assessment Quarterly*, 14(2):160–176, 2017. URL https://doi.org/10.1080/15434303.2016.1162793. [p123]

D. N. Lawley. On problems connected with item selection and test construction. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 61(3):273–287, 1943. URL https://doi.org/10.1017/S0080454100006282. [p124]

X. Li, J. Zhang, and H.-h. Chang. Look-ahead content balancing method in variable-length computerized classification testing. *British Journal of Mathematical and Statistical Psychology*, 73(1):88–108, 2020. [p123]

S.-C. Ma, H.-H. Wang, and T.-W. Chien. A new technique to measure online bullying: online computerized adaptive testing. *Annals of general psychiatry*, 16(1):26, 2017. URL https://doi.org/10.1186/s12991-017-0149-z. [p123]

D. Magis. A note on the equivalence between observed and expected information functions with polytomous irt models. *Journal of Educational and Behavioral Statistics*, 40(1):96–105, 2015. URL https://doi.org/10.3102/1076998614558122. [p124]

D. Magis and J. R. Barrada. Computerized adaptive testing with r: Recent updates of the package catr. *Journal of Statistical Software*, 76(1):1–19, 2017. URL https://doi.org/10.18637/jss.v076.c01. [p123]

D. Magis, G. Raîche, et al. Random generation of response patterns under computerized adaptive testing with the r package catr. *Journal of Statistical Software*, 48(8):1–31, 2012. [p123]

P. Michel, K. Baumstarck, A. Loundou, B. Ghattas, P. Auquier, and L. Boyer. Computerized adaptive testing with decision regression trees: an alternative to item response theory for quality of life measurement in multiple sclerosis. *Patient preference and adherence*, 12:1043–1053, 2018. URL https://doi.org/10.2147/PPA.S162206. [p123]

A. Mizumoto, Y. Sasao, and S. A. Webb. Developing and evaluating a computerized adaptive testing version of the word part levels test. *Language Testing*, 36(1):101–123, 2019. URL https://doi.org/10.1177/0265532217725776. [p123]

S. Nydick. *catIrt: An R Package for Simulating IRT-Based Computerized Adaptive Tests*, 2014. URL https://CRAN.R-project.org/package=catIrt. R package version 0.5-0. [p123]

M. W. Richardson. The relation between the difficulty and the differential validity of a test. *Psychometrika*, 1(2):33–49, 1936. URL https://doi.org/10.1007/BF02288003. [p124]

J. Rodríguez-Cuadrado, D. Delgado-Gómez, J. C. Laria, and S. Rodríguez-Cuadrado. Merged tree-cat: A fast method for building precise computerized adaptive tests based on decision trees. *Expert Systems with Applications*, 143:113066, 2020. [p123, 124, 125]

F. Samejima. Estimation of latent ability using a response pattern of graded scores. *Psychometrika monograph supplement*, 34(4, Pt. 2), 1969. [p124, 127]

L. R. Tucker. Maximum validity of a test with equivalent items. *Psychometrika*, 11(1):1–13, 1946. URL https://doi.org/10.1007/BF02288894. [p124]

M. Ueno. Adaptive testing based on bayesian decision theory. In *International Conference on Artificial Intelligence in Education*, pages 712–716. Springer, 2013. [p123]

M. Ueno and P. Songmuang. Computerized adaptive testing based on decision tree. In *2010 10th IEEE International Conference on Advanced Learning Technologies*, pages 191–193. IEEE, 2010. [p123]

W. J. Van der Linden and P. J. Pashley. Item selection and ability estimation in adaptive testing. In *Elements of adaptive testing*, pages 3–30. Springer, 2009. URL https://doi.org/10.1007/978-0-387-85461-8_1. [p123]

W. J. Veerkamp and M. P. Berger. Some new item selection criteria for adaptive testing. *Journal of Educational and Behavioral Statistics*, 22(2):203–226, 1997. URL https://doi.org/10.3102/10769986022002203. [p123]

D. J. Weiss. Computerized adaptive testing for effective and efficient measurement in counseling and education. *Measurement and Evaluation in Counseling and Development*, 37(2):70–84, 2004. URL https://doi.org/10.1080/07481756.2004.11909751. [p123]

H.-M. Wu, B.-C. Kuo, and S.-C. Wang. Computerized dynamic adaptive tests with immediately individualized feedback for primary school mathematics learning. *Educational Technology & Society*, 20(1):61–72, 2017. URL https://www.jstor.org/stable/jeductechsoci.20.1.61. [p123]

D. Yan, C. Lewis, and M. Stocking. Adaptive testing with regression trees in the presence of multidimensionality. *Journal of Educational and Behavioral Statistics*, 29(3):293–316, 2004. [p123]

X. Zhou and M. D. Reckase. Optimal item pool design for computerized adaptive tests with polytomous items using gpcm. *Psychological Test and Assessment Modeling*, 56(3):255, 2014. [p123]

Z. H. ZhuoKang and Y. Liu. Computational intelligence and intelligent systems. 2012. [p123]

Javier Rodríguez-Cuadrado
Universidad Carlos III de Madrid, Department of Statistics
Leganés
Spain
(https://orcid.org/0000-0003-1315-8977)
javierro@est-econ.uc3m.es

Juan C. Laria
Universidad Carlos III de Madrid, Department of Statistics
Leganés
Spain
(https://orcid.org/0000-0001-7734-9647)
jlaria@est-econ.uc3m.es

David Delgado-Gómez
Universidad Carlos III de Madrid, Department of Statistics
Leganés
Spain
(https://orcid.org/0000-0002-2976-2602)
ddelgado@est-econ.uc3m.es

A GUIded tour of Bayesian regression

by Andrés Ramírez–Hassan and Mateo Graciano-Londoño

Abstract This paper presents a Graphical User Interface (GUI) to carry out a Bayesian regression analysis in a very friendly environment without any programming skills (drag and drop). This paper is designed for teaching and applied purposes at an introductory level. Our GUI is based on an interactive web application using shiny and libraries from R. We carry out some applications to highlight the potential of our GUI for applied researchers and practitioners. In addition, the Help option in the main tap panel has an extended version of this paper, where we present the basic theory underlying all regression models that we developed in our GUI and more applications associated with each model.

Introduction

The main objective of this paper is to present an open source teaching Graphical User Interface (GUI) to implement Bayesian regression analysis using cross-sectional and longitudinal data.[1] We present a tutorial for implementing these models in our GUI and some applications. The Help option in the main tap panel exhibits an extended version of this paper where users can find more applications and the basic theoretical foundations of each model in our GUI. Therefore, practitioners and researchers can apply Bayesian regression analysis to understand its theoretical foundation without requiring programming skills. The latter seems to be a significant impediment to increasing the use of the Bayesian framework (Woodward, 2005; Karabatsos, 2016).

Table 1 shows the available graphical user interfaces for carrying out Bayesian regression analysis. **shinystan** (Stan Development Team, 2017) is a very flexible open source program, but users are required to have some programming skills. BugsXLA (Woodward, 2005) is open source but less flexible. However, users do not need to have programming skills. Bayesian regression: Nonparametric and parametric models (Karabatsos, 2016) is a very flexible and friendly GUI that is based on MATLAB Compiler for a 64-bit Windows computer. Its focus is on Bayesian nonparametric regressions, and it can be thought of for users who have mastered basic parametric models, such as the ones that we show in our GUI. On the other hand, MATLAB toolkit, Stata, and BayES are not open source.

We developed our GUI based on an interactive web application using **shiny** (Chang, 2018) and some libraries in R (R Core Team, 2018). The specific libraries and commands that are used in our GUI can be seen in Table 2. It has nine univariate models, four multivariate, three hierarchical longitudinal, Bayesian bootstrap, and six Bayesian model averaging frameworks. In addition, it gives basic summaries and diagnostics of the posterior chains, as well as the posterior chains themselves, and different plots, such as trace, autocorrelation, and densities. In terms of its flexibility and possibilities, our GUI lies between ShinyStan and BugsXLA: users are not required to have any programming skills, but it is not as advanced as Karabatsos (2016)'s software. However, our GUI can be run in any operating system. Our GUI, which we call BEsmarter,[2] is freely available at `https://github.com/besmarter/BSTApp`. Thus, users have access to all our code and datasets.

After this brief introduction, we present our GUI and how to use it in Section **Using BEsmarter**. Section **Applications** presents some empirical examples to illustrate the potential use of our GUI. Lastly, Section **Concluding remarks** presents some conclusions and future developments.

Using BEsmarter

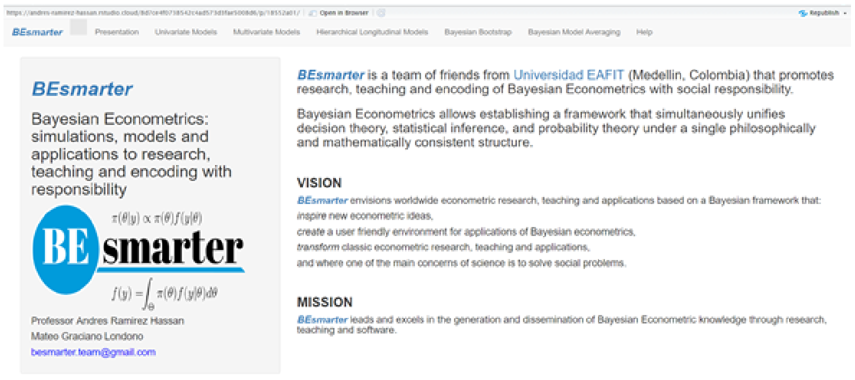
Simulated and applied datasets are in the folders *DataSim* (see Table 3 for details) and *DataApp* (see Table 4 for details), respectively. The former folder also includes the files that were used to simulate different processes so that the population parameters are available, and as a consequence, these files can be used as a pedagogical tool to show some statistical properties of the inferential frameworks available in our GUI. The latter folder contains the datasets used in our applications in Section **Applications**. Users should use these datasets as templates as a guide to the structure of their own datasets. Simply type **shiny::runGitHub("besmarter/BSTApp", launch.browser=T)** in the R package

[1]We used instrumental variables to identify causal effects when there are endogeneity issues in linear regression. This is a very common approach among econometricians in this situation. Otherwise, all regression approaches presented here are well known by statisticians.

[2]Bayesian econometrics: Simulations, models and applications to research, teaching, and encoding with responsibility.

console or any R code editor to run our GUI.[3]

After this, users can see a new window where a presentation of our research team is displayed. In addition, the top panel in Figure 1 shows the class of models that can be estimated in our GUI.



BEsmarter

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

$\pi(\theta|y) \propto \pi(\theta)f(y|\theta)$

BE smarter

$f(y) = \int_{\Theta} \pi(\theta)f(y|\theta)d\theta$

Professor Andres Ramirez Hassan
Mateo Graciano Londono
besmarter.team@gmail.com

BEsmarter is a team of friends from Universidad EAFIT (Medellin, Colombia) that promotes research, teaching and encoding of Bayesian Econometrics with social responsibility.

Bayesian Econometrics allows establishing a framework that simultaneously unifies decision theory, statistical inference, and probability theory under a single philosophically and mathematically consistent structure.

VISION

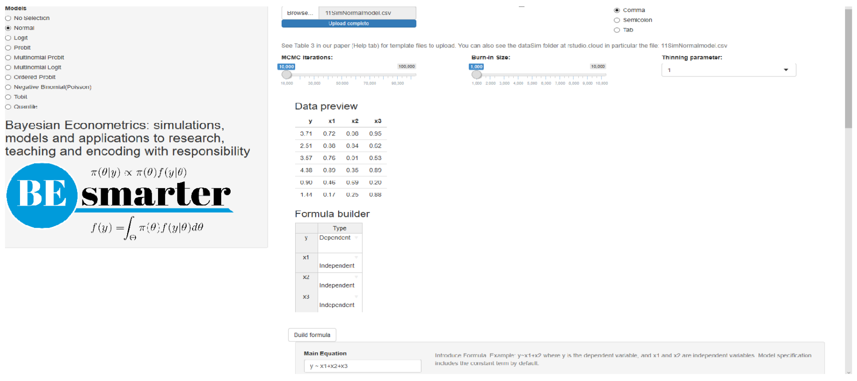
BEsmarter envisions worldwide econometric research, teaching and applications based on a Bayesian framework that:
inspire new econometric ideas,
create a user friendly environment for applications of Bayesian econometrics,
transform classic econometric research, teaching and applications,
and where one of the main concerns of science is to solve social problems.

MISSION

BEsmarter leads and excels in the generation and dissemination of Bayesian Econometric knowledge through research, teaching and software.

Figure 1: BEsmarter GUI

The selection indicates univariate models in that the radio button on the left hand side shows the specific models inside this generic class. In particular, users can see that the normal model is selected from inside the class of univariate models. See Figure 2.



Models
( ) No Selection
[✓ Normal]
( ) Logit
( ) Probit
( ) Multinomial Probit
( ) Multinomial Logit
( ) Ordered Probit
( ) Negative Binomial(Poisson)
( ) Tobit
( ) Quantile

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

$\pi(\theta|y) \propto \pi(\theta)f(y|\theta)$

BE smarter

$f(y) = \int_{\Theta} \pi(\theta)f(y|\theta)d\theta$

11SimNormalmodel.csv

[✓ Comma]
( ) Semicolon
( ) Tab

See Table 3 in our paper (Help tab) for template files to upload. You can also see the dataSim folder at rstudio.cloud in particular the file: 11SimNormalmodel.csv

MCMC Iterations: 10,000

Burn-in Size: 1,000

Thinning parameter: 1

Data preview

y	x1	x2	x3
3.71	0.72	0.08	0.95
2.51	0.08	0.84	0.62
3.57	0.76	0.01	0.53
4.38	0.89	0.35	0.89
0.90	0.46	0.59	0.20
1.44	0.17	0.25	0.88

Formula builder

	Type
y	Dependent
x1	Independent
x2	Independent
x3	Independent

Main Equation

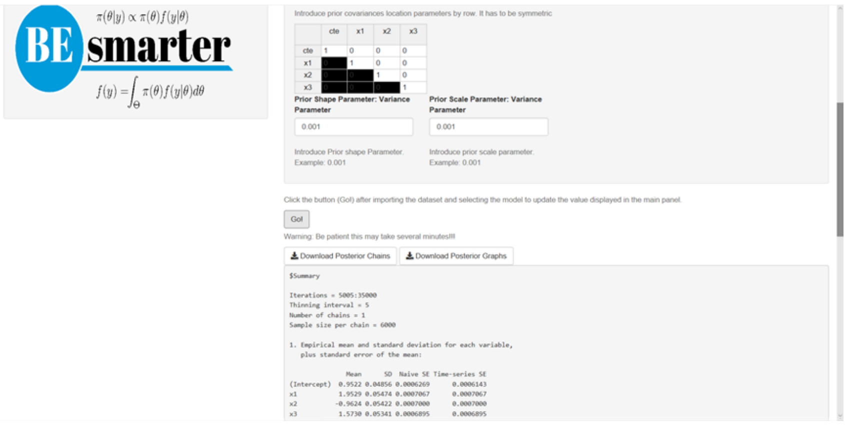
y ~ x1+x2+x3

Introduce Formula. Example: y~x1+x2 where y is the dependent variable, and x1 and x2 are independent variables. Model specification includes the constant term by default.

Figure 2: Univariate models: Specification

Then, the right-hand side panel displays a widget to upload the input dataset, which should be a csv file with headers in the first row. Users should also select the kind of separator used in the input file: comma, semicolon, or tab (use the folders *DataSim* and *DataApp* for the input file templates). Once users upload the dataset, they can see a data preview. Range sliders help to set the number of iterations of the MCMC and the amount of burn-in, and the thinning parameter can be selected as well (see online paper in Help tab for technical details). After this, users should specify the equation. This can be done with the formula builder, where users can select the dependent and the independent variables, and then click on the "Build formula" tab. Users can see in the "Main Equation" space the formula expressed in the format used by R (see Main equation box in Figure 2, $y \sim x1 + x2 + x3$). Users can modify this if necessary, for instance, including higher order or interaction terms. Other transformations are also allowed. This is done directly in the "Main Equation" space, taking into account that these extra terms should follow formula command structure (see https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/formula). Note that the class of univariate models includes the intercept by default, except ordered probit, where the specification has to do this explicitly. That is, ordered probit models do not admit an intercept for identification issues. Hence, users should write down specifically this fact ($y \sim x1 + x2 + x3 - 1$). Finally, users should define the hyperparameters of the prior. For instance, in the normal-inverse gamma model, these are the mean, covariance, shape, and scale (see Figure 3). However, users should take into account that our GUI has "non-informative" hyperparameters by default in all our modeling frameworks, so the last part is not a requirement.

[3]We strongly recommend to type this directly, rather than copy and paste. This is due to an issue with the quotation mark.


$\pi(\theta|y) \propto \pi(\theta)f(y|\theta)$
BE smarter
$f(y) = \int_{\Theta} \pi(\theta)f(y|\theta)d\theta$

Introduce prior covariances location parameters by row. It has to be symmetric

	cte	x1	x2	x3
cte	1	0	0	0
x1	0	1	0	0
x2	0	0	1	0
x3	0	0	0	1

Prior Shape Parameter: Variance Parameter

0.001

Introduce Prior shape Parameter. Example: 0.001

Prior Scale Parameter: Variance Parameter

0.001

Introduce prior scale parameter. Example: 0.001

Click the button (Go!) after importing the dataset and selecting the model to update the value displayed in the main panel.

Go!

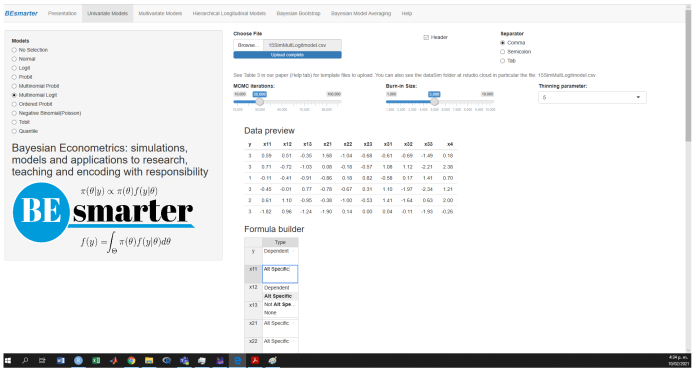
Warning: Be patient this may take several minutes!!!

```
$Summary

Iterations = 5005:35000
Thinning interval = 5
Number of chains = 1
Sample size per chain = 6000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

               Mean      SD  Naive SE Time-series SE
(Intercept)  0.9522 0.04856 0.0006269      0.0006143
x1           1.9529 0.05474 0.0007067      0.0007067
x2          -0.9624 0.05422 0.0007000      0.0007000
x3           1.5730 0.05341 0.0006895      0.0006895
```

Figure 3: Univariate models: Results

After this specification process, users should click the Go! button to initiate the estimation. Our GUI displays the summary statistics and convergence diagnostics after this process is finished (see Figure 3). There are also widgets to download posterior chains (csv file) and graphs (pdf and eps files). Note that the order of the coefficients in the results (summary, posterior chains, and graphs) is first for the location parameters and then for the scale parameters.

Multinomial models (probit and logit) require a dataset file to have the dependent variable in the first column, then alternative specific regressors (for instance, alternatives' prices), and finally, non-alternative regressors (for instance, income). The formula builder specifies the dependent variable and independent variables that are alternative specific and non-alternative specific. The specification also requires defining the base category, number of alternatives (this is also required in ordered probit), number of alternative specific regressors, and number of non-alternative regressors (see Figure 4). Multinomial logit also allows defining a tuning parameter, the number of degrees of freedom, in this case, for the Metropolis–Hastings algorithm (see online paper in Help tab for technical details). This is a feature in our GUI when the estimation of the models is based on the Metropolis–Hastings algorithm. The order of the coefficients in the results of these models is, first, the intercepts (cte_l appearing in the summary display, l-th alternative), then the non-alternative specific regressors (NAS_{jl} appearing in the summary display, l-th alternative and j-th non-alternative regressor), and lastly, the coefficients for the alternative specific regressors (AS_j appearing in the summary display, j-th alternative specific regressor). Note that the non-alternative specific regressors associated with the base category are equal to zero (they do not appear in the results). In addition, some coefficients of the main diagonal of the covariance matrix are constant due to identification issues in multinomial and multivariate probit models.



Models
- ○ No Selection
- ○ Normal
- ○ Logit
- ○ Probit
- ○ Multinomial Probit
- ● Multinomial Logit
- ○ Ordered Probit
- ○ Negative Binomial(Poisson)
- ○ Tobit
- ○ Quantile

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

Choose File 15SimMultLogitmodel.csv

☑ Header

Separator ● Comma ○ Semicolon ○ Tab

See Table 3 in our paper (Help tab) for template files to upload. You can also see the dataSim folder at rstudio.cloud in particular the file: 15SimMultLogitmodel.csv

MCMC iterations: 30,000 (10,000 – 100,000)

Burn-in Size: 5,000 (1,000 – 10,000)

Thinning parameter: 5

Data preview

y	x11	x12	x13	x21	x22	x23	x31	x32	x33	x4
3	0.59	0.51	-0.35	1.68	-1.04	-0.68	-0.61	-0.69	-1.49	0.18
3	0.71	-0.72	-1.03	0.08	-0.18	-0.57	1.08	1.12	-2.21	2.38
1	-0.11	-0.41	-0.91	-0.86	0.18	0.82	-0.58	0.17	1.41	0.70
3	-0.45	-0.01	0.77	-0.78	-0.67	0.31	1.10	-1.97	-2.34	1.21
2	0.61	1.10	-0.95	-0.38	-1.00	-0.53	1.41	-1.64	0.63	2.00
3	-1.82	0.96	-1.24	-1.90	0.14	0.00	0.04	-0.11	-1.93	-0.26

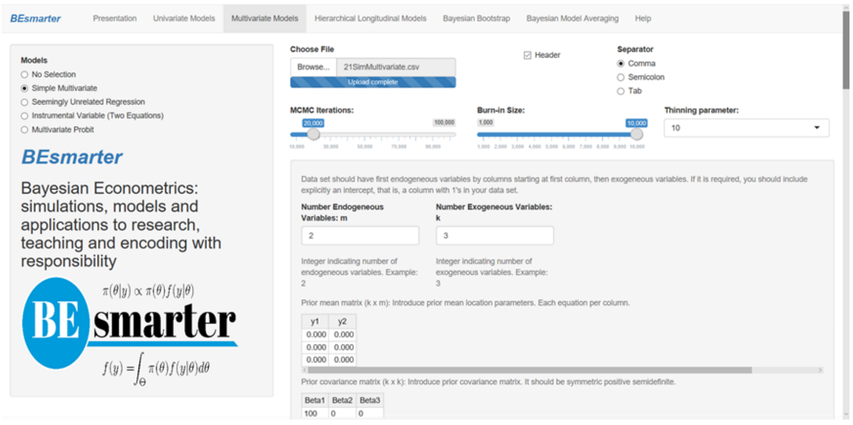
Formula builder

	Type
y	Dependent
x11	Alt Specific
x12	Dependent **Alt Specific**
x13	Not **Alt Spe**... None
x21	Alt Specific
x22	Alt Specific

Figure 4: Univariate models: Multinomial

In the case of the negative binomial model, users should set a dispersion parameter (α, see the negative binomial model). User should also set the censorship points and quantiles in the Tobit and quantile models, respectively.

Figure 5 displays the multivariate regression setting. In this case, the input file should have first the dependent variables and then the regressors. If there are intercepts in each equation, there should be a column of 1's after the dependent variables in the input file. The user also has to set the number of dependent variables, the number of regressors, if necessary include the intercept, and the values of the hyperparameters (see Figure 5).



Models
- No Selection
- [✓ Simple Multivariate]
- Seemingly Unrelated Regression
- Instrumental Variable (Two Equations)
- Multivariate Probit

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

Choose File: 21SimMultivariate.csv

[✓ Header]

Separator: [✓ Comma] Semicolon Tab

MCMC Iterations: 20,000

Burn-in Size: 10,000

Thinning parameter: 10

Data set should have first endogeneous variables by columns starting at first column, then exogeneous variables. If it is required, you should include explicitly an intercept, that is, a column with 1's in your data set.

Number Endogeneous Variables: m: 2

Integer indicating number of endogeneous variables. Example: 2

Number Exogeneous Variables: k: 3

Integer indicating number of exogeneous variables. Example: 3

Prior mean matrix (k x m): Introduce prior mean location parameters. Each equation per column.

y1	y2
0.000	0.000
0.000	0.000
0.000	0.000

Prior covariance matrix (k x k): Introduce prior covariance matrix. It should be symmetric positive semidefinite.

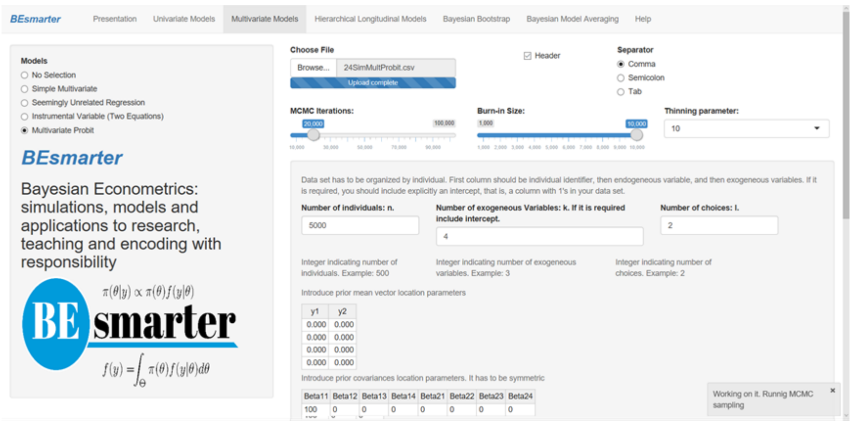
Beta1	Beta2	Beta3
100	0	0

Figure 5: Multivariate models: Simple multivariate

The input file in seemingly unrelated regressions should have first the dependent variables and then the regressors by equation, including the intercept in each equation if necessary (column of 1's). Users should define the number of dependent variables (equations) and the number of total regressors. That is, the sum of all regressors associated with the equation (if necessary include intercepts, each intercept is an additional regressor), and the number of regressors by equation (if necessary include the intercept). Users can also set the values of the hyperparameters if there is prior information.

The results of the simple multivariate and seemingly unrelated regressions show first the posterior location parameters by equation and then the posterior covariance matrix.

In the instrumental variable setting, users should specify the main equation and the instrumental equation. This setting includes intercepts by default. The first variable on the right-hand side in the main equation has to be the variable with endogeneity issues. In the instrumental equation box, the dependent variable is the variable with endogeneity issues as a function of the instruments. Users can also specify the values of the hyperparameters if they have prior information. The input file should have the dependent variable, the endogenous regressor, the instruments, and the exogenous regressors. The results first list the posterior estimates of the endogenous regressor, then the location parameters of the auxiliary regression (instrumental equation), and the location parameters of the exogenous regressors. Last is the posterior covariance matrix.



Models
- No Selection
- Simple Multivariate
- Seemingly Unrelated Regression
- Instrumental Variable (Two Equations)
- [✓ Multivariate Probit]

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

Choose File: 24SimMultProbit.csv

[✓ Header]

Separator: [✓ Comma] Semicolon Tab

MCMC Iterations: 20,000

Burn-in Size: 10,000

Thinning parameter: 10

Data set has to be organized by individual. First column should be individual identifier, then endogeneous variable, and then exogeneous variables. If it is required, you should include explicitly an intercept, that is, a column with 1's in your data set.

Number of individuals: n.: 5000

Integer indicating number of individuals. Example: 500

Number of exogeneous Variables: k. If it is required include intercept.: 4

Integer indicating number of exogeneous variables. Example: 3

Number of choices: l.: 2

Integer indicating number of choices. Example: 2

Introduce prior mean vector location parameters

y1	y2
0.000	0.000
0.000	0.000
0.000	0.000
0.000	0.000

Introduce prior covariances location parameters. It has to be symmetric

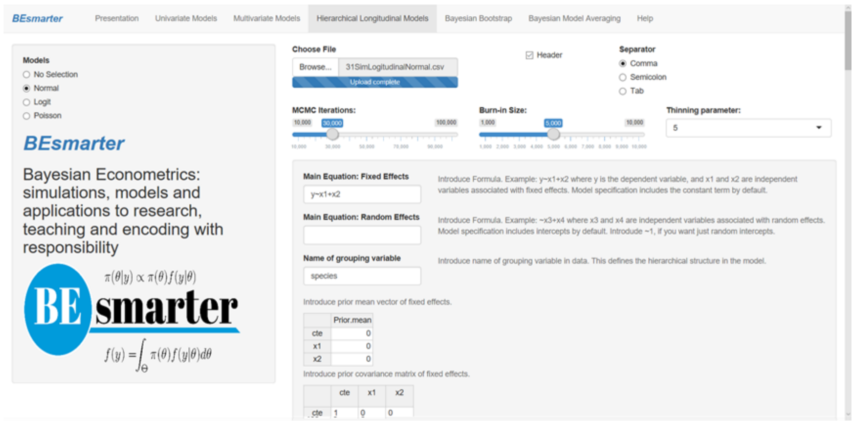
Beta11	Beta12	Beta13	Beta14	Beta21	Beta22	Beta23	Beta24
100	0	0	0	0	0	0	0

Working on it. Runnig MCMC sampling

Figure 6: Multivariate models: Multivariate probit

The multivariate probit model requires an input dataset ordered by unit. For instance, three choices imply repeating each unit three times. The first column has to be the identification of each unit; users should use ordered integers, then the dependent variable, just one vector, composed of 0's and 1's, then the regressors, which should include a column of 1's for the intercepts. Users should set the number of units, the number of regressors, and the number of choices (see Figure 6). The results first display the posterior location parameters by equation, and then the posterior covariance matrix.

The input files for hierarchical longitudinal models should have first the dependent variable, then the regressors, and a cross-sectional identifier ($i = 1, 2, \ldots, m$). It is not a requirement to have a balanced dataset: n_i can be different for each i. Users should specify the fixed part equation and the random part equation, both in R format. In case of only requiring random intercepts, do not introduce anything in the latter part (see Figure 7). Users should also type the name of the cross-sectional identifier variable. The results displayed and the posterior graphs are associated with the fixed effects and covariance matrix. However, users can download the posterior chains of all posterior estimates: fixed and random effects and covariance matrix.



Models
○ No Selection
● Normal
○ Logit
○ Poisson

Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

Choose File: 31SimLogitudinalNormal.csv

☑ Header

Separator
● Comma
○ Semicolon
○ Tab

MCMC Iterations: 30,000 (10,000 – 100,000)

Burn-in Size: 5,000 (1,000 – 10,000)

Thinning parameter: 5

Main Equation: Fixed Effects: y~x1+x2

Introduce Formula. Example: y~x1+x2 where y is the dependent variable, and x1 and x2 are independent variables associated with fixed effects. Model specification includes the constant term by default.

Main Equation: Random Effects:

Introduce Formula. Example: ~x3+x4 where x3 and x4 are independent variables associated with random effects. Model specification includes intercepts by default. Introdude ~1, if you want just random intercepts.

Name of grouping variable: species

Introduce name of grouping variable in data. This defines the hierarchical structure in the model.

Introduce prior mean vector of fixed effects.

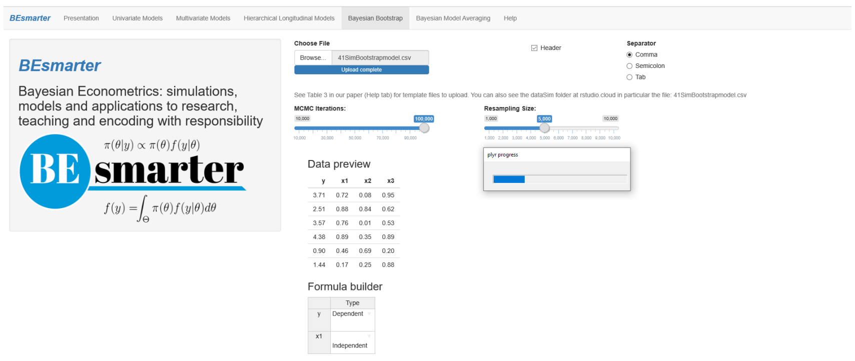
	Prior.mean
cte	0
x1	0
x2	0

Introduce prior covariance matrix of fixed effects.

	cte	x1	x2
cte	1	0	0

Figure 7: Hierarchical longitudinal models: Specification

Bayesian bootstrap only requires uploading a dataset, specifying the number of iterations of the MCMC, the resampling size, and the equation (see Figure 8). The input file has the same structure as the file used in the univariate normal model.



Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility

Choose File: 41SimBootstrapmodel.csv

☑ Header

Separator
● Comma
○ Semicolon
○ Tab

See Table 3 in our paper (Help tab) for template files to upload. You can also see the dataSim folder at rstudio.cloud in particular the file: 41SimBootstrapmodel.csv

MCMC Iterations: 100,000 (10,000 – 100,000)

Resampling Size: 5,000 (1,000 – 10,000)

Data preview

y	x1	x2	x3
3.71	0.72	0.08	0.95
2.51	0.88	0.84	0.62
3.57	0.76	0.01	0.53
4.38	0.89	0.35	0.89
0.90	0.46	0.69	0.20
1.44	0.17	0.25	0.88

Formula builder

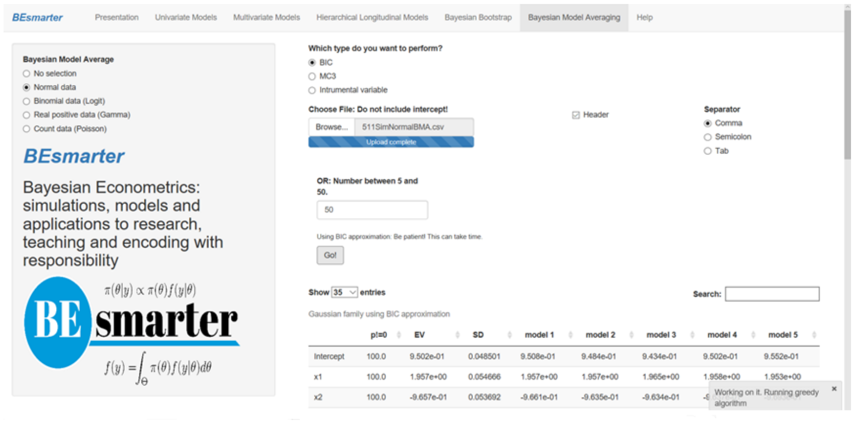
	Type
y	Dependent
x1	Independent

Figure 8: Bayesian bootstrap: Specification

Bayesian model averaging based on a Gaussian distribution can be carried out using the Bayesian Information Criterion (BIC) approximation, Markov chain Monte Carlo model composition (MC3), or instrumental variables (see Figure 9). The former two approaches require an input dataset where the first column is the dependent variable and then the potentially important regressors. Users should set the bandwidth model selection parameter (O_R) and the number of iterations for BIC and MC3, respectively. The results include the posterior inclusion probability ($p! = 0$), the expected value (EV), and the standard deviation (SD) of the coefficients associated with each regressor. The BIC framework also displays the most relevant models, including the number of regressors, the coefficient of determination (R^2), the BIC, and the posterior model probability. Users can download two csv files: *Best models* and *Descriptive statistics coefficients*. The former is a 0-1 matrix such that the columns are the regressors and the rows are the models; a 1 indicates the presence of a specific regressor in a specific model, 0 otherwise. Note that the last column of this file is the posterior model probability for each model (row). The latter file shows the posterior inclusion probabilities, expected values, and standard deviations associated with each regressor, taking into account the BMA procedure based on the best models.

Bayesian model averaging with endogeneity issues requires two input files. The first one has the dependent variable in the first column. The next columns are the regressors with endogeneity issues and then the exogenous regressors. The user should include a column of 1's if an intercept is required. The second input file has all the instruments. Users should also introduce the number of regressors with endogeneity issues (see Figure 10).

The results include the posterior inclusion probabilities and the expected values for each regressor. The user can find the results of the main equation and the auxiliary equations. Users can download csv


BEsmarter
Presentation
Univariate Models
Multivariate Models
Hierarchical Longitudinal Models
Bayesian Bootstrap
Bayesian Model Averaging
Help
BEsmarter
Bayesian Econometrics: simulations, models and applications to research, teaching and encoding with responsibility
BE smarter
$\pi(\theta|y) \propto \pi(\theta) f(y|\theta)$
$f(y) = \int_{\Theta} \pi(\theta) f(y|\theta) d\theta$
Browse...
Upload complete
Go!

Bayesian Model Average
No selection [✓ Normal data] Binomial data (Logit) Real positive data (Gamma) Count data (Poisson)

Which type do you want to perform?
[✓ BIC] MC3 Intrumental variable

Choose File: Do not include intercept! 511SimNormalBMA.csv
[✓ Header]
Separator: [✓ Comma] Semicolon Tab

OR: Number between 5 and 50.
50

Using BIC approximation: Be patient! This can take time.

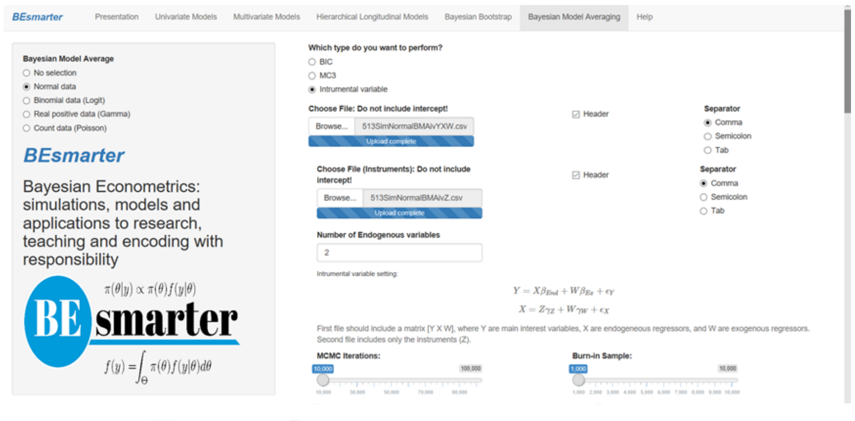
Show 35 entries
Search:

Gaussian family using BIC approximation

	p!=0	EV	SD	model 1	model 2	model 3	model 4	model 5
Intercept	100.0	9.502e-01	0.048501	9.508e-01	9.484e-01	9.434e-01	9.502e-01	9.552e-01
x1	100.0	1.957e+00	0.054666	1.957e+00	1.957e+00	1.965e+00	1.958e+00	1.953e+00
x2	100.0	-9.657e-01	0.053692	-9.661e-01	-9.635e-01	-9.634e-01	[illegible]	[illegible]

Working on it. Running greedy algorithm

Figure 9: Bayesian model averaging: Specification and results



Bayesian Model Average
No selection [✓ Normal data] Binomial data (Logit) Real positive data (Gamma) Count data (Poisson)

Which type do you want to perform?
BIC MC3 [✓ Intrumental variable]

Choose File: Do not include intercept! 513SimNormalBMAivYXW.csv
[✓ Header]
Separator: [✓ Comma] Semicolon Tab

Choose File (Instruments): Do not include intercept! 513SimNormalBMAivZ.csv
[✓ Header]
Separator: [✓ Comma] Semicolon Tab

Number of Endogenous variables
2

Intrumental variable setting:

$$Y = X\beta_{End} + W\beta_{Ex} + \epsilon_Y$$
$$X = Z\gamma_Z + W\gamma_W + \epsilon_X$$

First file should include a matrix [Y X W], where Y are main interest variables, X are endogeneous regressors, and W are exogenous regressors. Second file includes only the instruments (Z).

MCMC Iterations: 10,000 (range 10,000–100,000)

Burn-in Sample: 1,000 (range 1,000–10,000)

Figure 10: Bayesian model averaging: Instrumental variable specification

files of BMA results for both the second stage (main equation) and the first stage (auxiliary equations). In addition, users can download the posterior chains of the location parameters of the main equation, $\beta_l, l = 1, 2, \ldots, dim\{\boldsymbol{\beta}\}$, the location parameters of the auxiliary equations, $\gamma_{j,i}, j = 1, 2, \ldots, dim\{\boldsymbol{\beta}_s\}$, where $dim\{\boldsymbol{\beta}_s\}$ is the number of regressors with endogeneity issues, $i = 1, 2, \ldots, dim\{\gamma\}$, where $dim\{\gamma\}$ is the number of regressors in the auxiliary regressors (exogenous regressors + instruments), and the elements of the covariance matrix $\sigma_{j,k}$ (see online paper in Help tab for technical details).

Bayesian model averaging based on BIC approximation for non-linear models, logit, gamma, and Poisson requires an input dataset. The first column is the dependent variable and the other columns are the potentially relevant regressors. Users should specify the bandwidth model selection parameters, which are also referred to as Occam's window parameters (O_R and O_L). Our GUI displays the PIP ($p! = 0$), the expected value of the posterior coefficients (EV), and the standard deviation (SD). In addition, users can see the results associated with the models with the highest posterior model probabilities and download csv files with the results of specifications of the best model, and descriptive statistics of the posterior coefficients from the BMA procedure. These files are similar to the results of the BIC approximation of the Gaussian model.

User should also note that sometimes our GUI shuts down. In our experience, this is due to computational issues using the implicit commands that we call when estimating some models, for instance, computationally singular systems, missing values where TRUE/FALSE needed, L-BFGS-B needs finite values of "fn", NA/NaN/Inf values, or Error in backsolve. Sometimes these issues can be solved by adjusting the dataset, for instance, avoiding high levels of multicollinearity. In addition, users can identify these problems by checking the console of their rstudio cloud sections, where the specific folder/file where the issue happened is specified. In any case, we would appreciate your feedback to improve and enhance our GUI.

Applications

The main purpose of this section is to illustrate the potential of our GUI to carry out some applications. We encourage users to replicate these applications as we do not display in figures most of the results due to space limitations.[4] In addition, there are technical aspects that are covered in the online paper in Help tab of our GUI.

Univariate models

Continuous response: The market value of soccer players in Europe

We use the dataset *1ValueFootballPlayers.csv*, which was provided by Serna Rodríguez et al. (2018), to find the determinants of high-performance soccer players in the five most important national leagues in Europe.

The specification to enter in the main equation box is

$$\log(\text{Value}) \sim \text{Perf} + \text{Perf2} + \text{Age} + \text{Age2} + \text{NatTeam} + \text{Goals} + \text{Goals2} + \text{Exp} + \text{Exp2} + \text{Assists},$$

where Value is the market value in Euros (2017), Perf is a measure of performance, Age is the players' age in years, NatTem is an indicator variable that takes the value of 1 if the player has been on the national team, Goals is the number of goals scored by the player during his career, Exp is his experience in years, and Assists is the number of assists made by the player in the 2015–2016 season. All variables followed by a 2 are squared variables.

We initially assume that there are no censorship problems, the effect of the regressors are the same through the support of the dependent variable, and the dependent variable obeys normal distribution. So, we ran a normal-inverse gamma model using 30,000 MCMC iterations plus a burn-in equal to 5,000 and a thinning parameter equal to 1 using the default hyperparameters.

The results suggest that age, squared age, national team, goals, experience, and squared experience are relevant regressors. For instance, we found that the 2.5% and 97.5% percentiles of the posterior estimate associated with the variable Goals are 4.57e-03 and 1.82e-02. These values can be used to find the 95% symmetric credible interval. This means that there is a 0.95 probability that the population parameter lies in (4.57e-03, 1.82e-02), which would suggest that this variable is relevant to explain the market value of a soccer player.[5] We also found that the effect of having been on the national team has a 95% credible interval equal to (0.58, 1.04) with a median equal to 0.81. That is, an increase of the market value of the player of 124.8% ($\exp(0.81) - 1$) compared with a player that has not ever been on a national team. The posterior distribution of this variable can be seen in Figure 11. This graph is automatically generated by our GUI and can be downloaded in the zip file named *Posterior Graphs.csv*. However, we should take into account that the national team is the sixth variable. Remember that by default, the intercept is the first variable.

A good advantage of the Bayesian framework is that we can easily calculate the posterior distribution of functions of the parameter estimates,for instance, the age that maximizes the market value of a soccer player, OptAge $= -\frac{\beta_{Age}}{2\beta_{Age2}}$. We can estimate this using the posterior chains that can be downloaded from our GUI. This is in the file named *Posterior chains.csv*. We have that the mean value is equal to 24.31 years, and the 95% symmetric credible interval is (23.28, 25.36).

We can also see some convergence diagnostics from this application. In particular, the Geweke (1992) test indicates that there is no statistically significant difference at 5% between the first 10% of the posterior chains and the last 50% of the posterior chains. This is due to the fact that the absolute values of all the statistical tests are less than 1.96 (the value that defines the critical region in a normal distribution for a bilateral test at the 5% significance level). The Raftery and Lewis (1992) tests indicate dependence factors very close to 1 in all cases, and as a consequence ,lower than 5, which means a low level of autocorrelation of the posterior draws. Lastly, all the posterior chains passed the Heidelberger and Welch (1983) test, indicating that it seems that the posterior draws come from stationary distributions.

Let's assume that we only have the market value of soccer players whose value is greater than €1,000,000, which means that approximately 21.5% of our sample is censored. Estimating a normal-inverse gamma model without taking into account the censoring issue would mean inconsistent parameter estimates. For instance, we estimated a normal-inverse gamma model having a dependent

[4]Take into account that as inference in Bayesian models is based on simulation methods, results do not coincide 100%.

[5]Users should take into account that formal inference (hypothesis tests) in a Bayesian framework are based on Bayes factors.

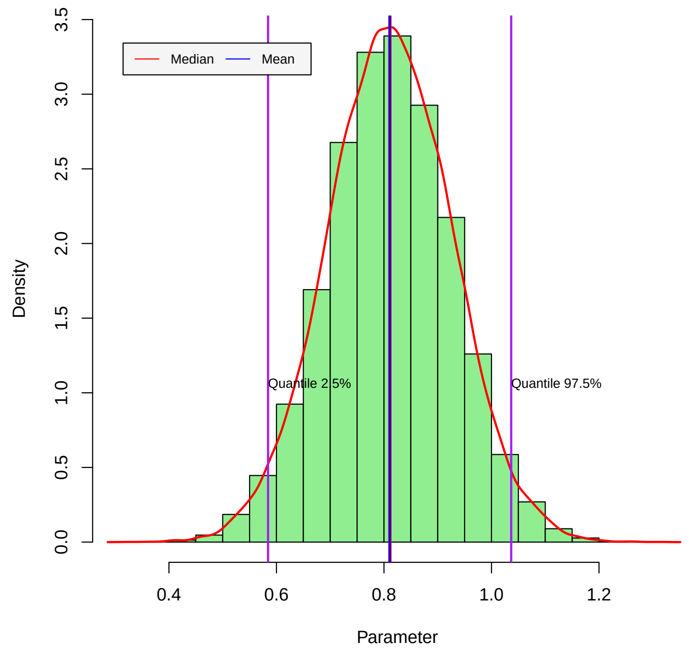


Figure 11: Posterior distribution: National team

variable log(ValueCens), which is the censored dependent variable, using the same setting as the baseline framework. We found that age, squared age, national team, goals, and experience are potentially relevant variables for predicting the market value, but this exercise suggests that squared experience is not relevant, a variable that was relevant in our previous estimation without censoring issues. Therefore, we estimated a Tobit model where log(ValueCens) is the dependent variable, which is left-censored at $\log(1,000,000) \approx 13.82$, with the same MCMC setting and hyperparameters as the baseline estimation. All convergence diagnostics seem good, and we got the same potentially relevant variables as in the baseline estimation, except for squared experience.

Now let us check if the marginal effects of the regressors are not constant over the support of the dependent variable. For instance, we want to check if the marginal effect of goals varies with the market value of the soccer player. So, we can estimate a Bayesian quantile regression. In particular, we estimated models at the 0.1, 0.5 (median), and 0.9 quantiles. We found that age, squared age, and national team are potentially relevant regressors to explain these quantiles. For instance, the age that maximizes the market value is approximately 24.5 years in all these three quantiles. However, goals are only relevant when we estimated the median model, which in general has better convergence diagnostics and narrower credible intervals. Observe that experience is not relevant in quantile regressions, whereas this variable is relevant in mean regressions.

Lastly, we carried out a Bayesian bootstrap, which means that we did not assume any particular distribution for the dependent variable. In particular, we set 20,000 iterations with a resample size equal to 1,000. We used the same specification as in the normal-inverse gamma model.

The results show the posterior mean estimates, the highest posterior density credible intervals at 95%, and some percentiles that can be used to obtain the 95% symmetric credible interval. It seems that age, squared age, national team, goals, experience, and squared experience are statistically significant variables to explain the market value of a soccer player. Observe that these variables were also relevant in the normal-inverse gamma model. For instance, the highest density and symmetric credible intervals for the national team are the same (0.61, 1.08). This is also similar to the 95% credible interval using the normal-inverse gamma model. All convergence statistics seem good, which suggests that the posterior draws come from stationary distributions.

Binary response: Determinants of hospitalization in Medellín

We use the dataset named *2HealthMed.csv*, which was provided by Ramírez Hassan et al. (2013). Our dependent variable is a binary indicator with a value equal to 1 if an individual was hospitalized in 2007, and 0 otherwise.

The equation to enter in the main equation box is

$$\text{Hosp} \sim \text{SHI} + \text{Female} + \text{Age} + \text{Age2} + \text{Est2} + \text{Est3} + \text{Fair} + \text{Good} + \text{Excellent},$$

where SHI is a binary variable equal to 1 if the individual is in a subsidized health care program and 0 otherwise, Female is an indicator of gender, Age in years, Age2 is squared age, Est2 and Est3 are indicators of socio-economic status, the reference is Est1, which is the lowest, and self perception of health status where bad is the reference.

We ran this application using a logit model with 30,000 MCMC iterations plus a burn-in equal to 10,000, a thinning parameter equal to 5, and a tuning parameter for the Metropolis–Hastings algorithm equal to 1.01. This implies an effective sample size equal to 6,000. Our results indicate that female and health status are relevant variables for hospitalization as their 95% credible intervals do not cross 0. Women have a higher probability of being hospitalized than do men, and people with bad self-perception of health conditions also have a higher probability of being hospitalized. Observe that we can use the posterior chains, which can be downloaded from our GUI, to obtain the posterior distributions of the marginal effects without extra computational burden.

We also carried out this application using the probit model with the baseline setting of the logit model. We got the same results regarding potentially relevant predictors.[6] However, the probit model does not require a tuning parameter in its MCMC algorithm, which in turn generates fewer autocorrelated chains.

Multivariate models

Continuous responses: The effect of institutions on per capita GDP

To illustrate the potential of our GUI to estimate multivariate models, we used the dataset provided by Acemoglu et al. (2001), who analyzed the effect of property rights on economic growth.

First of all, we used the dataset *5Institutions.csv* to estimate the following set of equations:

$$\log(\text{pcGDP95}_i) = \pi_0 + \pi_1 \log(\text{Mort}_i) + \pi_2\text{Africa} + \pi_3\text{Asia} + \pi_4\text{Other} + e_{1i}, \tag{1}$$

$$\text{PAER}_i = \gamma_0 + \gamma_1 \log(\text{Mort}_i) + e_{2i}, \tag{2}$$

where pcGDP95, PAER, and Mort are the per capita GDP in 1995, the average index of protection against expropriation between 1985 and 1995, and the settler mortality rate during the time of colonization. Africa, Asia, and Other are dummies for continents, with America as the baseline group.

As there are different sets of regressors in each equation, and we suspect there is a correlation between the stochastic errors of these two equations, we should estimate a seemingly unrelated regressions (SUR) model.

We should take into account that there are two equations: the first one has five regressors, including the intercept, and the second equation has two regressors (intercept plus the mortality rate). We used default values for the hyperparameters. This implies "vague" prior information, and hence an "objective" Bayesian approach.

We set 10,000 MCMC iterations plus 1,000 burn-in iterations and a thinning parameter equal to 1. It seems that this setting gives posterior chains that converge to stationary distributions. All stationary tests do not reject the null hypothesis of "stationarity," and the mixing properties look good (dependence factors close to 1, autocorrelation, and trace plots seem to indicate no autocorrelation).

The most important parameters are the effect of the mortality rate on gross domestic product and property rights. Their 95% credible intervals are (-0.67, -0.29) and (-0.85, -0.35), respectively (second and seventh parameters). This suggests that the settler mortality rate during the time of colonization is negatively associated with economic growth and property rights. In addition, the 95% credible interval of the covariance between the stochastic errors of these two equations is (0.33, 0.88), which suggests that there is statistically significant evidence of a correlation between the equations.

The previous set of equations can be considered as a restricted reduced form system, where the coefficients of the continents are set equal to 0 in the property rights equation. We can think in the following system of structural equations as producing the previous, but unrestricted, reduced form system,

[6]Remember that in this model our GUI displays the posterior results according to the order in the equation.

$$\log(\text{pcGDP95}_i) = \beta_0 + \beta_1 \text{PAER}_i + \beta_2 \text{Africa} + \beta_3 \text{Asia} + \beta_4 \text{Other} + u_{1i}, \tag{3}$$

$$\text{PAER}_i = \alpha_0 + \alpha_1 \log(\text{pcGDP95}_i) + \alpha_2 \log(\text{Mort}_i) + u_{2i}. \tag{4}$$

We used the file *4Institutions.csv*, which has the structure to estimate multivariate Bayesian regressions using our GUI, to identify the causal effect of property rights on per capita GDP. In particular, we use the same MCMC and hyperparameters setting as in the previous exercise to obtain the posterior estimates of the reduced system without imposing zero restrictions on the effect of continents on property rights. The structural parameter β_1 is equal to π_1/γ_1.[7] We used the posterior draws automatically generated by our GUI to obtain the posterior chain of this structural parameter, which are the causal effects that Acemoglu et al. (2001) wanted to identify. The 95% credible interval is (0.56, 2.93), the posterior mean value is 1.12, and the median value is 0.98. If we estimate a multivariate system without taking into account the dummy variables associated with the continents, the causal effect has a 95% credible interval (0.68, 1.43) with posterior mean and median values equal to 0.94 and 0.97, respectively. Observe that the length of the second interval is shorter than the first. This is because the dummy variables of the continents are not statistically relevant for the property rights equation. As a consequence, the former estimation is less efficient.

Observe that we also obtain the posterior draws of the covariance matrix of these two reduced form equations from our GUI. All the convergence diagnostics indicate that the posterior draws (location and scale parameters) seem to come from stationary distributions.

Another way to identify the causal effect of property rights on per capita GDP is using instrumental variables. Therefore, we used the file *6Institutions.csv* to estimate Equation 3 using the mortality rate as an instrument for property rights. The equation to enter in the main equation box is

$$\text{logpcGDP95} \sim \text{PAER} + \text{Africa} + \text{Asia} + \text{Other},$$

and the equation to enter in the instrumental equation box is

$$\text{PAER} \sim \text{logMort}.$$

We used 20,000 MCMC iterations plus a burn-in equal to 5,000 and a thinning parameter equal to 5. So, the effective length of the posterior draws is 4,000. Using the default hyperparameters, the 95% credible interval of the coefficient associated with the endogenous variable, which is the first to be displayed in our descriptive and diagnostic statistics, is (0.55, 1.21), and the mean value is equal to 0.82. So, this is the effect of property rights on per capita GDP. Our GUI display next the posterior results associated with the instrumental equation, there we obtained a 95% credible interval equal to (-0.83, -0.35) for the effect of the mortality rate on the property rights. This suggests that the instrument is not weak. Then, we obtained the posterior results for the exogenous regressors in the main equation, which suggests that Africa and Asia dummies variables have negative effects on per capita GDP. Finally, we got the posterior estimates for the covariance matrix, which suggest that there is a negative covariance between the GDP equation and PAER equation, the 95% credible interval is (-1.50, -0.26).

All posterior draws seem to come from stationary distributions. However, there are high levels of autocorrelation in some posterior chains, as suggested by the dependence factors and posterior plots.

Hierarchical longitudinal models

Normal model: The relation between productivity and public investment

We used the dataset named *8PublicCap.csv* used by Ramírez Hassan (2017) to analyze the relation between public investment and gross state product in the setting of a spatial panel dataset consisting of 48 US states from 1970 to 1986. In particular, the specification to type into the main equation box of fixed effects is

$$\log(\text{gsp}) \sim \log(\text{pcap}) + \log(\text{pc}) + \log(\text{emp}) + \text{unemp},$$

where gsp in the gross state product, pcap is public capital, and pc is private capital all in US$, emp is employment (people), and unemp is the unemployment rate in percentage.

[7] Substituting Equation 4 into Equation 3, and comparing it with Equation 1 yields $\pi_1 = \frac{\beta_1\alpha_2}{1-\beta_1\alpha_1}$. Solving for the PAER as a function of the exogenous regressors in the structural system, and comparing it with Equation 2 yields $\gamma_1 = \frac{\alpha_2}{1-\beta_1\alpha_1}$. Observe one needs independent equations ($\beta_1\alpha_1 \neq 1$) and the exclusion restriction ($\alpha_2 \neq 0$).

We left the main equation box of random effects empty as we assumed that the unobserved heterogeneity is not associated with any particular regressors. This means that we control for the unobserved heterogeneity using just the constant terms. The variable which identifies the units is id.

We ran this application using 10,000 MCMC iterations plus a burn-in equal to 5,000 iterations and a thinning parameter equal to 1. We also used the default values for the hyperparameters of the prior distributions. It seems that all posterior draws come from stationary distributions as suggested by the diagnostics and posterior plots.

The 95% symmetric credible intervals for public capital, private capital, employment, and unemployment, are (-2.54e-02, -2.06e-02), (2.92e-01, 2.96e-01), (7.62e-01, 7.67e-01), and (-5.47e-03, -5.31e-03), respectively. The posterior mean elasticity estimate of public capital to gsp is -0.023. That is, an increase of 1% in public capital means a 0.023% decrease in gross state product. The posterior mean estimates of private capital and employment elasticities are 0.294 and 0.765, respectively. In addition, a 1% increase in the unemployment rate means a decrease of 0.54% in gsp. It seems that all these variables are statistically relevant. In addition, the posterior mean estimates of the variance associated with the unobserved heterogeneity and stochastic errors are 1.06e-01 and 1.45e-03. We obtained the posterior chain of the proportion of the variance associated with the unobserved heterogeneity (see Figure 12). The 95% symmetric credible interval is (0.98, 0.99) for this proportion. That is, unobserved heterogeneity is very important to explain the total variability.

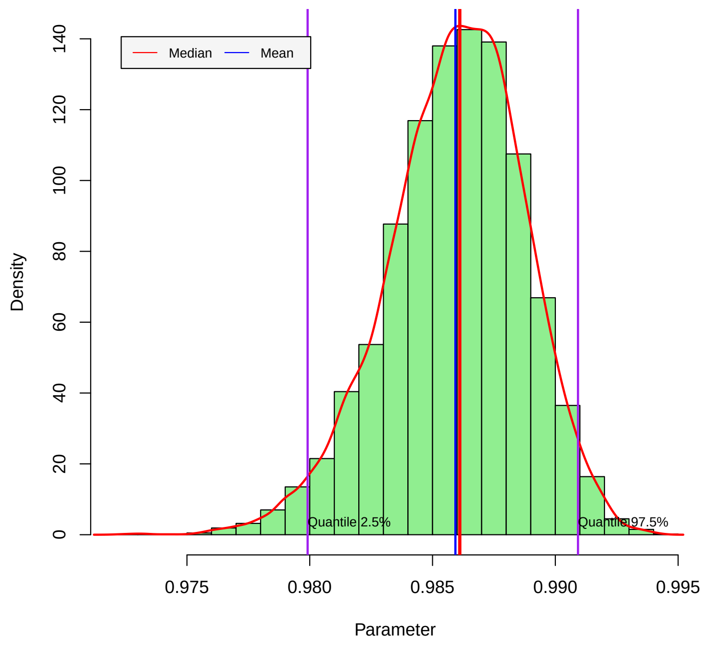


Figure 12: Posterior distribution: Proportion of variance associated with unobserved heterogeneity

Bayesian model averaging

Continuous response: Determinants of export diversification

We used the dataset provided by Jetter and Ramírez Hassan (2015) to analyze the determinants of export diversification. The dataset named *10ExportDiversificationHHI.csv* contains information about 36 potential determinants of export diversification measured using the Herfindahl–Hirschman Index (avghhi) for 104 countries (see Jetter and Ramírez Hassan (2015) for details). This setting implies 68.7 billion models (2^{36}).

We implemented three Bayesian Model Average (BMA) strategies: Bayesian Information Criterion approximation (BIC), Markov chain Monte Carlo model composition (MC^3), and instrumental variable (IVBMA). The former takes into account possible endogeneity between export diversification and gross domestic product.

Regarding BMA using the BIC approximation, we set 50 (default value) for OR. This parameter defines the number of best models to take into account in our BMA strategy. We obtained a table where we can see the posterior inclusion probability (PIP), expected value, standard deviation, and posterior mean estimates associated with the best models for each variable. The best models are defined using posterior model probabilities, which appear at the bottom of the table, where we also see the number of variables associated with each model as well as the coefficients of determination and BIC values. Our GUI also produces two csv files. The first one is *Best Models.csv*, where we have by row the best models and the variables by columns, a 1 indicates the presence of the specific variable in the model's specification, and a 0 its absence. The last column in this file is the posterior model probability. The second one is *Descriptive Statistics.csv*, where we see the posterior inclusion probability, expected value, and standard deviation of each variable.

Following Kass and Raftery (1995)'s suggestions, we found that there is very strong evidence that being a former colony of Portugal, the total net primary enrollment and the total natural resources rents as a percentage of GDP are determinants of export diversification. Their expected values are 0.15, -0.006 and 0.008, respectively, which means that there are negative effects of having been a colony of Portugal and of having natural resources on export diversification. Recall that higher values of HHI indicate less diversification.

We also ran this application using the MC3 strategy with 10,000 MCMC iterations. We got results similar to those with the BIC approximation.

We estimated an instrumental variable BMA to take into account possible endogeneity between export diversification and GDP using 20,000 MCMC iterations plus a burn-in equal to 5,000, where there is one endogenous variable (GDP). In particular, we used the files *11ExportDiversificationHHI.csv* and *12ExportDiversificationHHIInstr.csv*. The first file has the dependent variable in the first column (avhhhi) followed by the endogenous variable (avglgdpcap), the constant term (a column of 1's), and exogenous regressors. The second file has the instrumental variables, which are geographical, cultural, and colonial factors.

Our GUI first displays the outcomes of the second stage equation (main equation) and then the first stage equation (instrumental equation). We can download three csv files: *BMA Results First Stage.csv*, *BMA Results Second Stage.csv*, and *Posterior chains.csv*. The first two files have the same structure: posterior inclusion probabilities and expected values. We can see from these files that educational levels and governance performance are the most important variables to foster gross domestic product (PIP=100), primary enrollment fosters export diversification (PIP=79.5), whereas natural resources discourage it (PIP=97.1). The latter file has the posterior draws where the name *beta* is associated with the variables in the main equation (second stage), and *gamma* is associated with the instrumental variable equation (first stage). Lastly, we have the posterior draws of the covariance matrix of the stochastic errors in the first and second-stage equations. We have a 95% symmetric credible interval equal to (-0.014, 0.024), suggesting that there are no endogeneity issues.

Concluding remarks

The Bayesian statistical framework has become very popular among scientists since the computational revolution in the 1990s. In particular, computationally burdensome procedures such as Markov chain Monte Carlo algorithms can be easily implemented nowadays. However, most of the open source software to apply these procedures requires programming skills. This may be one reason why the Bayesian framework is not very popular among applied researchers and practitioners. In this paper, we introduced a graphical user interface to implement Bayesian regression analysis under different frameworks, explaining the basic theory so that users can understand the basic principles of Bayesian statistics and apply them easily. Our objective has been to increase the popularity of the Bayesian statistical framework among applied researchers and practitioners.

Name	Language	Models	Open source
ShinyStan	R+Stan	MCMC Implementation*	Yes
Bayesian regression: NP&P	MATLAB Compiler	Bayesian infinite-mixture regression Bayesian normal regression Hierarchical linear regression Binary regression Ordered regression Censoring regression Quantile regression Survival regression Density estimation Variable selection (spike-and-slab)	Yes
BugsXLA	OpenBUGS + Excel	Normal linear models GLM: Binomial GLM: Poisson GLM: Survival GLM: Multivariate categorical data Normal linear mixed Generalized linear mixed Bayesian variable selection Robust models	Yes
MATLAB toolkit: E&E+	MATLAB	Linear Regression Regression with non-spherical errors Regime switch regression Regression with restricted parameters Seemingly unrelated regression (SUR) Vector AutoRegression (VAR) Instrumental variable Probit and logit Tobit Model Panel Data Analysis Stochastic search variable selection Highest posterior density (HPD) region Marginal likelihood of linear regression	No
Stata	Stata	MCMC implementation*	No
BayES	C++	Simple linear model Random-effects Random-coefficients Stochastic frontiers Inefficiency-effects Random-effects stochastic frontiers Dynamic stochastic frontier Probit and logit Random-effects probit and logit Multinomial probit and logit Ordered probit and logit Poisson and negative-binomial Type I Tobit Type II Tobit Seemingly unrelated regressions (SUR) Vector Autoregressive (VAR)	No

*User should define prior and likelihood.
+Toolkit on econometrics and economics teaching.

Table 1: Graphical user interfaces to perform Bayesian regression analysis.

Univariate models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMCregress	Martin et al. (2018)
Logit	MCMCpack	MCMClogit	Martin et al. (2018)
Probit	bayesm	rbprobitGibbs	Rossi (2017)
Multinomial(Mixed) Probit	bayesm	rmnpGibbs	Rossi (2017)
Multinomial(Mixed) Logit	bayesm	rmnlIndepMetrop	Rossi (2017)
Ordered Probit	bayesm	rordprobitGibbs	Rossi (2017)
Negative Binomial(Poisson)	bayesm	rnegbinRw	Rossi (2017)
Tobit	MCMCpack	MCMCtobit	Martin et al. (2018)
Quantile	MCMCpack	MCMCquantreg	Martin et al. (2018)
Multivariate models			
Model	**Library**	**Command**	**Reference**
Multivariate	bayesm	rmultireg	Rossi (2017)
Seemingly Unrelated Regression	bayesm	rsurGibbs	Rossi (2017)
Instrumental Variable	bayesm	rivGibbs	Rossi (2017)
Bivariate Probit	bayesm	rmvpGibbs	Rossi (2017)
Hierarchical longitudinal models			
Model	**Library**	**Command**	**Reference**
Normal	MCMCpack	MCMChregress	Martin et al. (2018)
Logit	MCMCpack	MCMChlogit	Martin et al. (2018)
Poisson	MCMCpack	MCMChpoisson	Martin et al. (2018)
Bayesian Bootstrap			
Model	**Library**	**Command**	**Reference**
Bayesian bootstrap	bayesboot	bayesboot	Baath (2018)
Bayesian model averaging			
Model	**Library**	**Command**	**Reference**
Normal (BIC)	BMA	bic.glm	Raftery et al. (2012)
Normal (MC^3)	BMA	MC3.REG	Raftery et al. (2012)
Normal (instrumental variables)	ivbma	ivbma	Lenkoski et al. (2013)
Logit (BIC)	BMA	bic.glm	Raftery et al. (2012)
Gamma (BIC)	BMA	bic.glm	Raftery et al. (2012)
Poisson (BIC)	BMA	bic.glm	Raftery et al. (2012)
Diagnostics			
Diagnostic	**Library**	**Command**	**Reference**
Trace plot	coda	traceplot	Plummer et al. (2016)
Autocorrelation plot	coda	autocorr.plot	Plummer et al. (2016)
Geweke test	coda	geweke.diag	Plummer et al. (2016)
Raftery & Lewis test	coda	raftery.diag	Plummer et al. (2016)
Heidelberger & Welch test	coda	heidel.diag	Plummer et al. (2016)

Table 2: Libraries and commands in BEsmarter GUI.

Univariate models		
Model	**Data set file**	**Data set simulation**
Normal	11SimNormalmodel.csv	11SimNormal.R
Logit	12SimLogitmodel.csv	12SimLogit
Probit	13SimProbitmodel.csv	13SimProbit.R
Multinomial(Mixed) Probit	14SimMultProbmodel.csv	14SimMultinomialProbit.R
Multinomial(Mixed) Logit	15SimMultLogitmodel.csv	15SimMultinomialLogit.R
Ordered Probit	16SimOrderedProbitmodel.csv	16SimOrderedProbit.R
Negative Binomial(Poisson)	17SimNegBinmodel.csv	17SimNegBin.R
Tobit	18SimTobitmodel.csv	18SimTobit.R
Quantile	19SimQuantilemodel.csv	19SimQuantile.R
Multivariate models		
Model	**Data set file**	**Data set simulation**
Multivariate	21SimMultivariate.csv	21SimMultReg.R
Seemingly Unrelated Regression	22SimSUR.csv	22SimSUR.R
Instrumental Variable	23SimIV.csv	23SimIV.R
Bivariate Probit	24SimMultProbit.csv	24SimMultProbit.R
Hierarchical longitudinal models		
Model	**Data set file**	**Data set simulation**
Normal	31SimLogitudinalNormal.csv	31SimLogitudinalNormal.R
Logit	32SimLogitudinalLogit.csv	32SimLogitudinalLogit.R
Poisson	33SimLogitudinalPoisson.csv	33SimLogitudinalPoisson.R
Bayesian Bootstrap		
Model	**Data set file**	**Data set simulation**
Bayesian bootstrap	41SimBootstrapmodel.csv	41SimBootstrapmodel.R
Bayesian model averaging		
Model	**Data set file**	**Data set simulation**
Normal (BIC)	511SimNormalBMA.csv	511SimNormalBMA.R
Normal (MC^3)	512SimNormalBMA.csv	512SimNormalBMA.R
Normal (instrumental variables)	513SimNormalBMAivYXW.csv 513SimNormalBMAivZ.csv	513SimNormalBMAiv.R
Logit (BIC)	52SimLogitBMA.csv	52SimLogitBMA.R
Gamma (BIC)	53SimGammaBMA.csv	53SimGammaBMA.R
Poisson (BIC)	53SimPoissonBMA.csv	53SimPoissonBMA.R

Table 3: Data sets templates in folder *DataSim*.

Univariate models		
Model	**Data set file**	**Dependent variable**
Normal	1ValueFootballPlayers.csv	$\log(\text{Value})$
Logit	2HealthMed.csv	Hosp
Probit	2HealthMed.csv	Hosp
Multinomial(Mixed) Probit	Fishing.csv	mode
Multinomial(Mixed) Logit	Fishing.csv	mode
Ordered Probit	2HealthMed.csv	MedVisPrevOr
Negative Binomial(Poisson)	2HealthMed.csv	MedVisPrev
Tobit	1ValueFootballPlayers.csv	$\log(\text{ValueCens})$
Quantile	1ValueFootballPlayers.csv	$\log(\text{Value})$
Multivariate models		
Model	**Data set file**	**Dependent variable**
Multivariate	4Institutions.csv	logpcGDP95 and PAER
Seemingly Unrelated Regression	5Institutions.csv	logpcGDP95 and PAER
Instrumental Variable	6Institutions.csv	logpcGDP95 and PAER
Bivariate Probit	7HealthMed.csv	$y = [\text{Hosp SHI}]'$
Hierarchical longitudinal models		
Model	**Data set file**	**Dependent variable**
Normal	8PublicCap.csv	$\log(\text{gsp})$
Logit	9VisitDoc.csv	DocVis
Poisson	9VisitDoc.csv	DocNum
Bayesian Bootstrap		
Model	**Data set file**	**Dependent variable**
Bayesian bootstrap	1ValueFootballPlayers.csv	$\log(\text{Value})$
Bayesian model averaging		
Model	**Data set file**	**Dependent variable**
Normal (BIC)	10ExportDiversificationHHI.csv	avghhi
Normal (MC^3)	10ExportDiversificationHHI.csv	avghhi
Normal (instrumental variables)	11ExportDiversificationHHI.csv 12ExportDiversificationHHIInstr.csv	avghhi and avglgdpcap
Logit (BIC)	13InternetMed.csv	internet
Gamma (BIC)	14ValueFootballPlayers.csv	Ln market value
Poisson (BIC)	15Fertile2.csv	ceb

Table 4: Real data sets in folder *DataApp*.

Bibliography

D. Acemoglu, S. Johnson, and J. Robinson. The colonial origins of comparative development: An empirical investigation. *The American Economic Review*, 91(5):1369–1401, 2001. [p143, 144]

R. Baath. *Package bayesboot*, 2018. URL https://CRAN.R-project.org/package=bayesboot. [p148]

W. Chang. *Web Application Framework for R: Package shiny*. R Studio, 2018. URL http://shiny.rstudio.com/. [p135]

J. Geweke. *Bayesian Statistics*, chapter Evaluating the accuracy of sampling-based approaches to calculating posterior moments. Clarendon Press, Oxford, UK., 1992. [p141]

P. Heidelberger and P. D. Welch. Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6):1109–1144, 1983. [p141]

M. Jetter and A. Ramírez Hassan. Want export diversification? educate the kids first. *Economic Inquiry*, 53(4):1765–1782, 2015. [p145]

G. Karabatsos. A menu-driven software package of bayesian nonparametric (and parametric) mixed models for regression analysis and density estimation. *Behavior Research Methods*, 49:335–362, 2016. [p135]

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995. [p146]

A. Lenkoski, A. Karl, and A. Neudecker. *Package ivbma*, 2013. URL https://CRAN.R-project.org/package=ivbma. [p148]

A. D. Martin, K. M. Quinn, and J. H. Park. *Package MCMCpack*, 2018. URL https://CRAN.R-project.org/package=MCMCpack. [p148]

M. Plummer, N. Best, K. Cowles, K. Vines, D. Sarkar, D. Bates, R. Almond, and A. Magnusson. *Output Analysis and Diagnostics for MCMC*, 2016. URL https://CRAN.R-project.org/package=coda. [p148]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL http://www.R-project.org/. [p135]

A. Raftery and S. Lewis. One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science*, 7:493–497, 1992. [p141]

A. Raftery, J. Hoeting, C. Volinsky, I. Painter, and K. Y. Yeung. *Package BMA*, 2012. URL https://CRAN.R-project.org/package=BMA. [p148]

A. Ramírez Hassan. The interplay between the Bayesian and frequentist approaches: a general nesting spatial panel data model. *Spatial Economic Analysis*, 12(1):92–112, 2017. [p144]

A. Ramírez Hassan, J. Cardona Jiménez, and R. Cadavid Montoya. The impact of subsidized health insurance on the poor in Colombia: Evaluating the case of Medellín. *Economia Aplicada*, 17(4): 543–556, 2013. [p142]

P. Rossi. *Package bayesm*, 2017. URL https://CRAN.R-project.org/package=bayesm. [p148]

M. Serna Rodríguez, A. Ramírez Hassan, and A. Coad. Uncovering value drivers of high performance soccer players. *Journal of Sport Economics*, Accepted manuscript, 2018. [p141]

Stan Development Team. shinystan: Interactive visual and numerical diagnostics and posterior analysis for bayesian models., 2017. URL http://mc-stan.org/. R package version 2.3.0. [p135]

P. Woodward. BugsXLA: Bayes for the common man. *Journal of Statistical Software*, 14(5):1–18, 2005. [p135]

Andrés Ramírez–Hassan
Department of Economics
School of Economics and Finance
Universidad EAFIT
Cra. 49, Medellín, Antioquia, Colombia
ORCID: 0000-0002-0467-7903

e-mail: aramir21@eafit.edu.co
URL: www.besmarter-team.org

Mateo Graciano-Londoño
Quantitative Analyst Department
Risk Viceprecidency
Protección SA
Cra. 49, 63 10, Medellín, Antioquia, Colombia.
e-mail: mgracian@proteccion.com.co
URL: www.besmarter-team.org

StratigrapheR: Concepts for Litholog Generation in R

by Sébastien Wouters, Anne-Christine Da Silva, Frédéric Boulvain and Xavier Devleeschouwer

Abstract The StratigrapheR package proposes new concepts for the generation of lithological logs, or lithologs, in R. The generation of lithologs in a scripting environment opens new opportunities for the processing and analysis of stratified geological data. Among the new concepts presented: new plotting and data processing methodologies, new general R functions, and computer-oriented data conventions are provided. The package structure allows for these new concepts to be further improved, which can be done independently by any R user. The current limitations of the package are highlighted, along with the limitations in R for geological data processing, to help identify the best paths for improvements.

Introduction

StratigrapheR is a package implemented in the open-source programming environment R. **StratigrapheR** endeavors to explore new concepts to process stratified geological data. These concepts are provided to answer a major difficulty posed by such data; namely a large amount of field observations of varied nature, sometimes localized and small-scale, can carry information on large-scale processes. Visualizing the relevant observations all at once is therefore difficult. The usual answer to this problem in successions of stratified rocks is to report observations in a schematic form: the lithological log, or litholog (e.g., Fig. 1). The litholog is an essential tool in sedimentology and stratigraphy and proves to be equally invaluable in other fields such as volcanology, igneous petrology, or paleontology. Ideally, any data contained in a litholog should be available in a reproducible form. Therefore, the challenge at hand is what we would call "from art to useful data"; how can we best extract and/or process the information contained in a litholog, designed to be as visually informative as possible (see again Fig. 1).

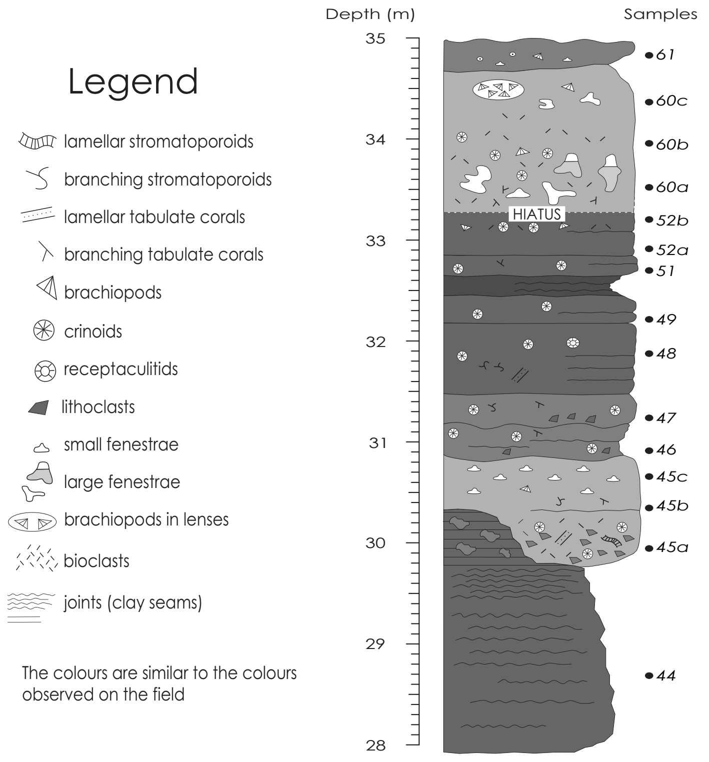


Figure 1: Example of a computer-drawn litholog of calcareous rocks, modified from Humblet and Boulvain (2000) using vector graphics software (e.g., Inkscape, CorelDRAW, or Adobe Illustrator).

Lithologs can be hand-drawn, computer-drawn, or generated via ad hoc software tools. Drawn figures can have unlimited precision and personalization. They are, however, time-consuming to produce and ill-adapted for the extraction of data for further numerical analysis. Moreover, any modification to drawn lithologs has to be performed manually. Ad hoc software tools such as the open-source SedLog program (Zervas et al., 2009) or the **SDAR** R package (Ortiz et al., 2019) propose a solution to such short-comings by generating lithologs from geological data provided in a text format. The most common text format is the American Standard Code for Information Interchange -ASCII-. ASCII is used for the SedLog data format and for the Log ASCII Standard -LAS- (Heslop et al., 1999). The latter (LAS) is the data format used by the **SDAR** package.

A major advantage of ad hoc software tools is that any change in the data can automatically lead to an update in the display of the litholog. However, ad hoc software tools only permit a certain amount of personalization. The graphical output of ad hoc software tools, which can be obtained in vector graphics format (e.g., in the Scalable Vector Graphics [SVG] format), has to be post-processed in vector graphics software to add elements that are not supported by the data format. In SedLog, for instance, to add plots next to the litholog (i.e., to visualize quantified analytical values and their relation to the lithological features) the plots need to be generated separately and then added manually along the litholog in vector graphics software. SedLog does permit a certain amount of personalization, but only for the lithological symbology, by giving the user the option of adding self-made symbology (e.g., to show the position of paleontological or sedimentological features). In the **SDAR** package, the only data automatically displayable are Gamma Ray spectrometry values, and the symbology (for lithology, fossils, etc.) cannot be personalized.

Generally speaking, the graphical style of the lithologs generated by ad hoc software tools is difficult to personalize entirely. To do so, each functionality has to be modular, which is better done in a scripting language such as R. Yet, the **SDAR** package, although coded in R, is not modular. All the plotting is made using a single function. Any personalization feature would need to be explicitly coded into that function, which would be a never-ending task. Moreover, ad hoc software tools are difficult to update and improve. Adding new functionalities or maintaining the software for compatibility with new operating systems, for example, usually falls on the shoulders of the developers of that software.

The **StratigrapheR** package is presented here as a new mean to generate lithologs. It provides a complementary approach to the existing methodologies and circumvents the aforementioned problems. **StratigrapheR** is designed not around a specific data format but on general tools able to deal with different formats. This opens a way of processing the geological data through a scripting language which has a large potential to evolve. **StratigrapheR** shows that symbiosis between automation and personalization is achievable for litholog generation. As it stands, the package does not meet the "from art to useful data" challenge entirely. However, it is a proof of concept showing that, despite the artistic nature of lithologs, they can be based on usable digital data, or that conversely, usable data can be extracted from drawn lithologs.

StratigrapheR is coded in R, which disposes of automated package checks (Wickham, 2015) and is itself updated regularly. This is one mechanism against the inevitable obsolescence of the functionalities. Similarly, as the **StratigrapheR** package is structured in distinct basic functions, implementing new functionalities (or updating existing ones) can be done more easily by any user. Furthermore, the processing of geological data, whether to generate lithologs or for any other procedure (among others plotting proxies, applying moving averages on these proxies, or performing spectral analysis), can directly be performed in R (see, for instance, the **paleotree** package for paleontology (Bapst, 2012), the **IsoplotR** package for geochronology (Vermeesch, 2018), or the **hht** (Bowman and Lees, 2013), **astrochron** (Meyers, 2014), **biwavelet** (Gouhier et al., 2019) and **DecomposeR** (Wouters, 2020) packages for spectral analysis). This means that the entire data treatment and visualization could be performed in a single scripting environment: R.

The main concepts for the use of **StratigrapheR** are presented in this paper. The current limitations of **StratigrapheR** and R for the processing of geological data are also highlighted to give an idea of the obstacles that the future developers will need to overcome to make R a better tool for geological data processing. Throughout the paper, examples are provided on how to make lithologs and how to process geological data. They can be run in R (you can download R here); the current version of **StratigrapheR** (1.2.3) works only on R 4.0 or higher versions. A GitHub repository is available at https://github.com/sewouter/StratigrapheR, where outside users can suggest improvements and provide feedback. The free RStudio interface is advised to use **StratigrapheR** in the R environment. The **StratigrapheR** package can be installed by typing:

```
install.packages("StratigrapheR")
```

To be used, the **StratigrapheR** package has to be loaded each time R or RStudio are opened, via the following code:

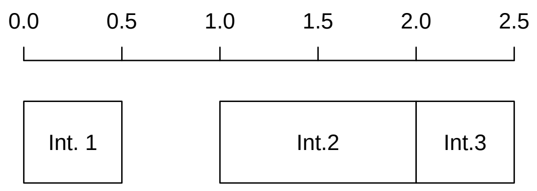
```
library(StratigrapheR)
```

Data importation and processing

Data of any form can easily be imported using basic R functions, such as `read.table()` or `readLines()` for text files. Excel files can be downloaded using, for instance, the `read.xlsx()` function from the **xlsx** package (Dragulescu and Arendt, 2020). We advise putting any tabular data into data frame form (i.e., a table), which can be done via the `data.frame()` function.

As stratigraphic data can be found in an interval form (e.g., a specific strata between 25 and 30 m in a record, or the Jurassic between ca. 200 and ca. 145 million years ago), a formal scheme to deal with such data is provided: the 'lim' object (named after the `xlim` and `ylim` parameters that define the boundaries of plots in common R graphical functions) and a suite of functions that are associated to lim objects. The idea is to set a logical data format for intervals and to be able to manipulate these intervals in R. The lim objects are made via the `as.lim()` function by providing boundaries in the form of the `l` and `r` arguments, which respectively stand for left and right boundaries. The actual order of the boundaries is irrelevant to avoid unnecessary data cleaning (which is the reason why 'left' and 'right' were chosen as a convention rather than 'up' and 'down'). Each interval can be identified using the `id` argument. Providing the upper and lower boundaries allows taking gaps into account in lithologs, contrary to simply providing the thickness of layers (also called beds). Whether the boundaries are included in the interval can be determined via the `b` argument, which defines the boundary rules. This is an abstract feature, especially for geology purposes, because it is usually of negligible importance whether the infinitesimal position of a boundary is included in a given interval. However, taking this into account is critical to explicitly describe the behavior of intervals. This can be used, for instance, to assign an interval to a sample located at the common boundary between two intervals that do not overlap otherwise. By providing a boundary rule, it can be explicitly assigned to only one of the two intervals, none of them, or both of them. The boundary rule is expressed by characters, and can be set to `"[]"` (or `"closed"`) to include both boundary points, `"]["` (or `"()"`, and `"open"`) to exclude both boundary points, `"[["` (or `"[)"`, `"right-open"` and `"left-closed"`) to include only the left boundary point, and `"]]"` (or `"(]"`, `"left-open"`,and `"right-closed"`) to include only the right boundary point. The left element (e.g., the `[` of `"[]"`) stands for the left boundary (not necessarily the lowest one), while the right element (e.g., the `]` of `"[]"`) stands for the right boundary (not necessarily the highest one). We illustrate how to visualize intervals with the following code (note: graphics generated by code in the article are shown directly after the code that generates them):

```
interval <- as.lim(l = c(0,1,2), r = c(0.5,2,2.5),   # Make a lim object
                   id = c("Int. 1","Int.2","Int.3"))

interval  # print what is in the lim object
#> $l
#> [1] 0 1 2
#> $r
#> [1] 0.5 2.0 2.5
#> $id
#> [1] "Int. 1" "Int.2"  "Int.3"
#> $b
#> [1] "[]" "[]" "[]"

# Visualization of the lim object
plot.new()
plot.window(ylim = c(-0.5, 2.5), xlim = c(0, 2.5))
axis(3, pos = 1.5, las = 1)

infobar(ymin = 0, ymax = 1, xmin = interval$l, xmax = interval$r,
        labels = c(interval$id), srt = 0)
```



Functions are provided to characterize the relationships of intervals with each other: `are.lim.nonunique()` checks whether the intervals are of non-zero thickness (e.g., unlike [1,1]), `are.lim.nonadjacent()` checks if the intervals do not share any adjacent boundaries, and `are.lim.distinct()` checks whether the intervals are not overlapping. The `simp.lim()` function is provided to merge adjacent and/or overlapping intervals having identical IDs. The `flip.lim()` function is provided to find the complementary intervals of a set of intervals (i.e., the gaps). The `mid.lim()` function provides a way to define intervals in between data points. If all different intervals are strictly non-overlapping for all values (for instance, the intervals [0,20[and [20, 100] are non-overlapping and therefore 20 is uniquely represented by the second interval), the `in.lim()` function can be used to find which values belong to which respective intervals. Typically, such functions can be used to craft stratigraphic intervals (such as magnetochrons or stages) and determine the beds or samples that are in or outside them.

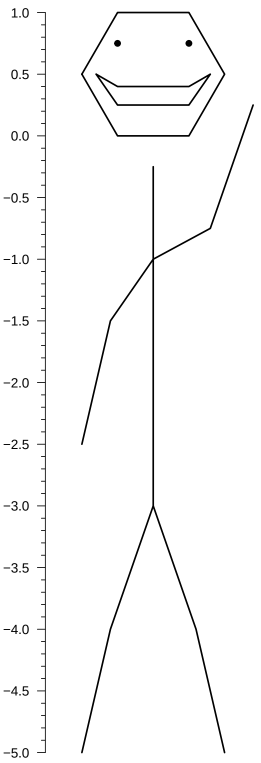
General plotting considerations

The first challenge when plotting a litholog is its size: a litholog needs to be detailed at a small scale (typically at the centimeter scale for high precision) while spanning the entirety of a record (up to hundreds of meters). This makes lithologs sometimes quite extended. This is problematic considering that the classical R graphic window is not adapted to visualize anything exceeding the size of the computer screen. To remedy this problem, the `pdfDisplay()` function is here introduced, which draws plots directly in a PDF (Portable Document File) document and opens it in the computer's default PDF reader. This PDF document can have any size desired by the user, and therefore, allows visualizing all the details of a very long litholog. This is illustrated by the code here below, which draws a vertically standing stickman. Depending on the screen, this vertical stickman could be difficult to visualize without `pdfDisplay()`.

To avoid having to close the PDF reader at each change of the plot (as most PDF readers do not permit changes to the PDF file while it is displayed), each new PDF can be named differently: each new document version will have its name be followed by '_(i)', where i is the version number (e.g., test_(1).pdf, test_(2).pdf, etc.). This practice is here referred as *tracking* the version number. It is noteworthy to cite the free Sumatra PDF software, which is available for Windows operating systems, and lets PDF files be modified while still displaying them without the trick of having to change the file name. In that case, the tracking of the version numbers can be canceled by setting the `track` parameter to `FALSE`. PDF files generated by `pdfDisplay()` can easily be imported into vector graphics software. The `pdfDisplay()` function also allows for the direct generation of an SVG file. `pdfDisplay()` is a wrapper of the more basic `pdf()` function (i.e., its code is based on the `pdf()` function); other PDF generating functions could be used interchangeably.

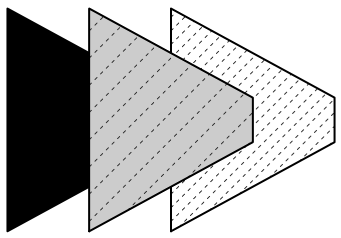
To make plots starting from an empty background, we advise using the `plot.new()` and `plot.window()` functions, which are of lower level (i.e., more basic) than the `plot()` function. They are used to create a completely empty plotting environment in which to add the litholog. To add axes, the `axis()` function is a versatile tool, which can be replaced by the `minorAxis()` function provided in **StratigrapheR** to add minor axis ticks. The `minorAxis()` function is itself a wrapper of the `axis()` function.

```
graphical_function <- function() # Graphical function needed by pdfDisplay
{
   opar <- par()$mar # Save initial graphical parameters
   par(mar = c(0,3,0,1)) # Change the margins of the plot

   plot.new() # Open a new plot
   plot.window(xlim = c(-0.2, 1.2), ylim = c(-5, 1)) # Define plot coordinates
   minorAxis(2, at.maj = seq(-5, 1, 0.5), n = 5, las = 1) # Add axis
   points(c(0.25, 0.75), c(0.75, 0.75), pch = 19)
   polygon(c(0.1, 0.25, 0.75, 0.9, 0.75, 0.25, NA,
             0, 0.25, 0.75, 1, 0.75, 0.25),
           c(0.5, 0.25, 0.25, 0.5, 0.4, 0.4, NA,
             0.5, 0, 0, 0.5, 1, 1), lwd = 2)
   lines(x = c(0.5, 0.5, NA, 0, 0.2, 0.5, 0.8, 1, NA,
               0, 0.2, 0.5, 0.9, 1.2),
         y = c(-0.25, -3, NA, -5, -4, -3, -4, -5, NA,
               -2.5, -1.5, -1, -0.75, 0.25), lwd = 2)

   par(mar = opar) # Restore initial graphical parameters
}

pdfDisplay(graphical_function(),"graphical_function", width = 3.5, height = 10)
```

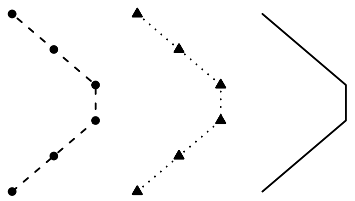


Adding every element of the lithologs symbology uses a very basic data format for polylines and polygons, which are respectively drawn using the `multigons()` and `multilines()` functions. These novel functions allow precise control of graphical parameters when drawing multiple polygons and polylines. These functions require an identification argument `i`, similar for each point of a single polygon or polyline, and the `x` and `y` coordinates of each point. The following code shows the use of the `multigons()` and `multilines()` functions:

```
i <- c(rep("A1",6), rep("A2",6), rep("A3",6)) # Polygon IDs
x <- c(1,2,3,3,2,1,2,3,4,4,3,2,3,4,5,5,4,3)   # x coordinates
y <- c(1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6)   # y coordinates

plot.new()
plot.window(xlim = c(0,6), ylim = c(0,7))

multigons(i, x, y,
          front = "A2", # This gets the polygon A2 in front of all others
          density = c(NA, 5, 10),  # Shading line density
          scol = "grey20", # Shading line color; one value means all polygons
                           # are subject to this graphical parameter
          col = c("black", "grey80", "white"), # Background color
          lwd = 2, # Width of border lines
          slty = 2, slwd = 1) # Shading lines type and width
```

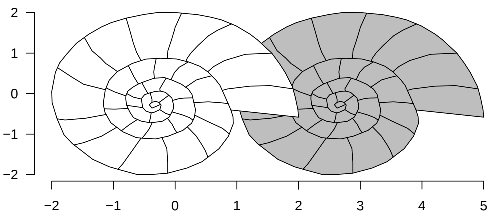


```
i <- c(rep("A1",6), rep("A2",6), rep("A3",6)) # Lines IDs
x <- c(1,2,3,3,2,1,4,5,6,6,5,4,7,8,9,9,8,7)   # x coordinates
y <- c(1,2,3,4,5,6,1,2,3,4,5,6,1,2,3,4,5,6)   # y coordinates

plot.new()
plot.window(xlim = c(0,10), ylim = c(0,7))

multilines(i, x, y,
           j = c("A3", "A1", "A2"),  # j controls the order of the graphical
                                     # parameters applied to each named line:
           lty =  c(1,2,3), lwd = 2, # e.g., lty = 1 (solid line) is applied
                                     # to "A3", the line at the right
           type = c("l", "o", "o"),
           pch = c(NA,21,24), cex = 1, bg = "black")
```



The pointsvg() function is provided in **StratigrapheR** to import groups of polygons and polylines drawn in vector graphics software, under specific conditions: firstly, the drawing needs to be in SVG format; secondly, the pointsvg() function is only able to identify objects of class "line", "rect", "polygon", and "polyline" in the SVG file. The reason for this is that only these types of objects are simple lines, and polygons made of nodes linked together by straight lines. This means that pointsvg() is not able to recognize all the objects present in an SVG file. Furthermore, pointsvg() only identifies the coordinates of each objects, regroups them into separate polygons and polyline objects, and in which order to plot them. All other graphical parameters, such as color or line thickness, are not taken into account. These parameters have to be specified in the drawing functions. Objects obtained using pointsvg() on SVG files can be added using the framesvg() or centresvg() functions, which respectively add the object within a given frame or center the object on a given point.

```
svg.file.directory <- tempfile(fileext = ".svg")     # Creates temporary file
writeLines(example.ammonite.svg, svg.file.directory) # Writes svg in the file

ammonite.drawing <- pointsvg(file = svg.file.directory) # Read svg

plot.new()
plot.window(xlim = c(-2, 5), ylim = c(-2, 2))
axis(1)
axis(2, las = 2)

centresvg(ammonite.drawing,  # Object
          x = c(3,0), y = 0,  # Coordinates for centering
          xfac = 2, yfac = 2,  # Dimension stretching factors
          col = c("grey","white"))  # Graphical parameters
```



It should be noted that repetitions of the same SVG object can be generated by a single call of the `framesvg()` or `centersvg()` functions. This facilitates the automation of the litholog generation. Modifications of the SVG objects can also be accomplished using the `changesvg()` function, which enables, among other things, to change the order of plotting of the polylines and polygons, remove some of them, or invert the figure in x and/or y. The `framesvg()` or `centersvg()` can also output the drawing with modified coordinates, which can be plotted using `placesvg()` (see, for instance, the code of the last example).

Generating lithologs

The data to make lithologs can be provided in the form shown in Table 1.

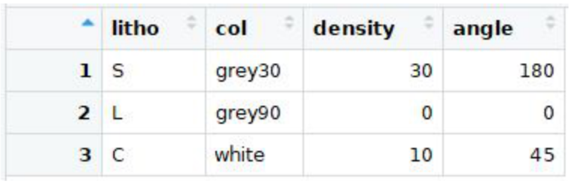
id	l	r	h	colour	litho
B1	0	1	3	grey	S
B2	1	3	4	grey	L
B3	3	4	5	black	C
B4	4	9	4	white	L
B5	9	11	4	white	L
...	...	...	...	...	...

Table 1: Example of a data frame (`bed.example` in **StratigrapheR**) providing information for each bed: id identifies each bed, l and r provide the boundaries, h the hardness, and the color is provided along with a code for lithology (S for shale, L for limestone, C for chert). The only strict convention is that l, r, and h need to be numerical values.

From such data, basic lithologs made of rectangles can be generated as a simple basis. They are the starting point for making more complicated lithologs in **StratigrapheR**. The coordinates of the points making up the rectangles can be computed through the `litholog()` function, which only needs the position of the boundaries of the beds, their 'hardness', and an ID. Text can be added to each bed using the `bedtext()` function, which can be used to include the ID or the name of the bed (e.g., id in Table 1).

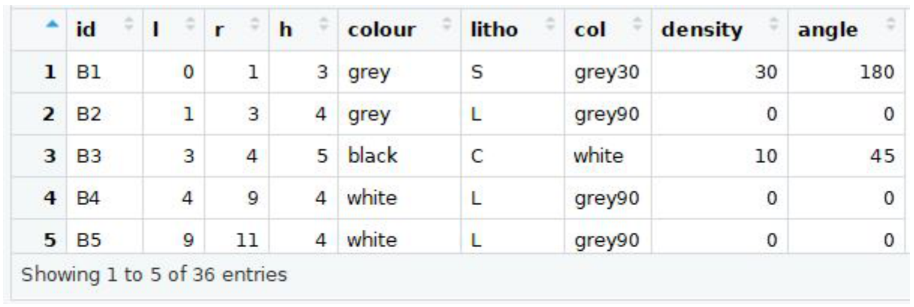
The output of the `litholog()` function can be provided to `multigons()` to draw the log. A symbology for different types of rocks (or any other information that the symbology is meant to provide) can be set up using the color fill and the shading. Providing a given symbology for each polygon is performed by joining the table containing the information about each bed to a table attributing symbology to rock type. We advise the use of the `left_join()` function in the **dplyr** package (Wickham et al., 2020) for this procedure.

```
basic.log <- litholog(l = bed.example$l,  # This creates a data table of
                      r = bed.example$r,  # rectangles coordinates for a
                      h = bed.example$h,  # basic litholog
                      i = bed.example$id)

legend <- data.frame(litho = c("S", "L", "C"),             # This creates a
                     col = c("grey30", "grey90", "white"), # data table for
                     density = c(30, 0,10),                # the symbology
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
View(legend)
```



	litho	col	density	angle
1	S	grey30	30	180
2	L	grey90	0	0
3	C	white	10	45

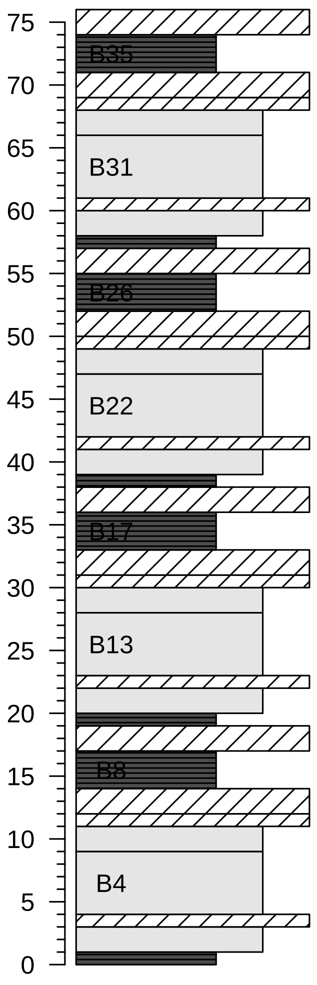
```
# left_join in the dplyr package merges the symbology with the table of beds:
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")

View(bed.legend)
```



	id	l	r	h	colour	litho	col	density	angle
1	B1	0	1	3	grey	S	grey30	30	180
2	B2	1	3	4	grey	L	grey90	0	0
3	B3	3	4	5	black	C	white	10	45
4	B4	4	9	4	white	L	grey90	0	0
5	B5	9	11	4	white	L	grey90	0	0

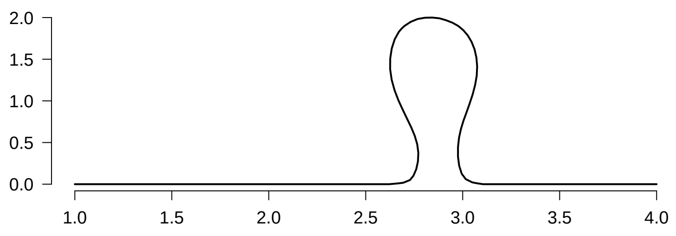
Showing 1 to 5 of 36 entries

```
plot.new()
plot.window(xlim = c(0,6), ylim = c(-1,77))
minorAxis(2, at.maj = seq(0, 75, 5), n = 5)

# Plotting of the polygons making the litholog,
# with corresponding symbology:
multigons(basic.log$i, x = basic.log$xy, y = basic.log$dt,
          col = bed.legend$col,
          density = bed.legend$density,
          angle = bed.legend$angle)

# Writing the name of beds, only in beds thick enough
bedtext(labels = bed.example$id, l = bed.example$l, r = bed.example$r,
        x = 0.5,  # x position where to center the text
        ymin = 3) # text is not added in beds thinner than ymin
```

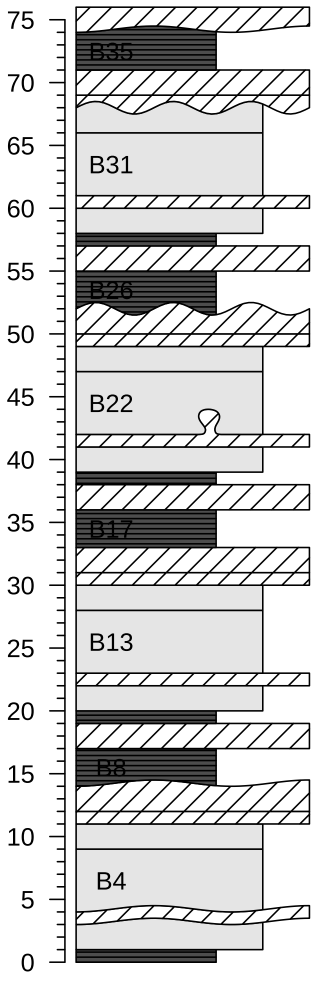


To add more complicated beds, the user can add SVG drawings instead of drawing the rectangles through `multigons()`, as shown earlier. This is, however, a time-consuming procedure as each bed has to be imported separately. The `weldlog()` function can be used to automate the personalization of bed boundaries. It needs to be provided as a polyline, either from R itself (e.g., a sinusoid) or from an SVG file.

```
# Code repeated from earlier examples ----
basic.log <- litholog(l = bed.example$l, r = bed.example$r,
                      h = bed.example$h, i = bed.example$id)
legend <- data.frame(litho = c("S", "L", "C"), density = c(30, 0,10),
                     col = c("grey30", "grey90", "white"),
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")
# ----

# Generation of the boundaries, either sinusoidal or from drawings ---
s1 <- sinpoint(5,0,0.5,nwave = 1.5)
s2 <- sinpoint(5,0,1,nwave = 3, phase = 0)
s3 <- framesvg(example.liquefaction, 1, 4, 0, 2, plot = FALSE, output = TRUE)

# Visualizing the s3 boundary, i.e., the liquefaction sedimentary feature ----
plot(s3$x, s3$y, cex.axis = 1.2, lwd = 2,
     type = "l", ylab = "", xlab = "", bty = "n", las = 1)
```



```
# Welding the boundaries to the basic litholog ----
final.log <- weldlog(log = basic.log,
                     dt = boundary.example$dt, # Position of the boundaries
                                               # to be changed
                     seg = list(s1 = s1, s2 = s2, s3 = s3), # list of segments
                     j = c("s1","s1","s1","s3", # Attributing the segments to
                           "s2","s2","s1"),     # the respective bed boundaries
                                                # to  be changed
                     warn = F)

# Visualizing the resulting litholog (similarly to earlier code) ----
plot.new()
plot.window(xlim = c(0,6), ylim = c(-1,77))
minorAxis(2, at.maj = seq(0, 75, 5), n = 5, las = 1)

multigons(final.log$i, x = final.log$xy, y = final.log$dt,
          col = bed.legend$col,
          density = bed.legend$density,
          angle = bed.legend$angle)

bedtext(labels = bed.example$id, l = bed.example$l, r = bed.example$r,
        x = 0.75, ymin = 3)
```

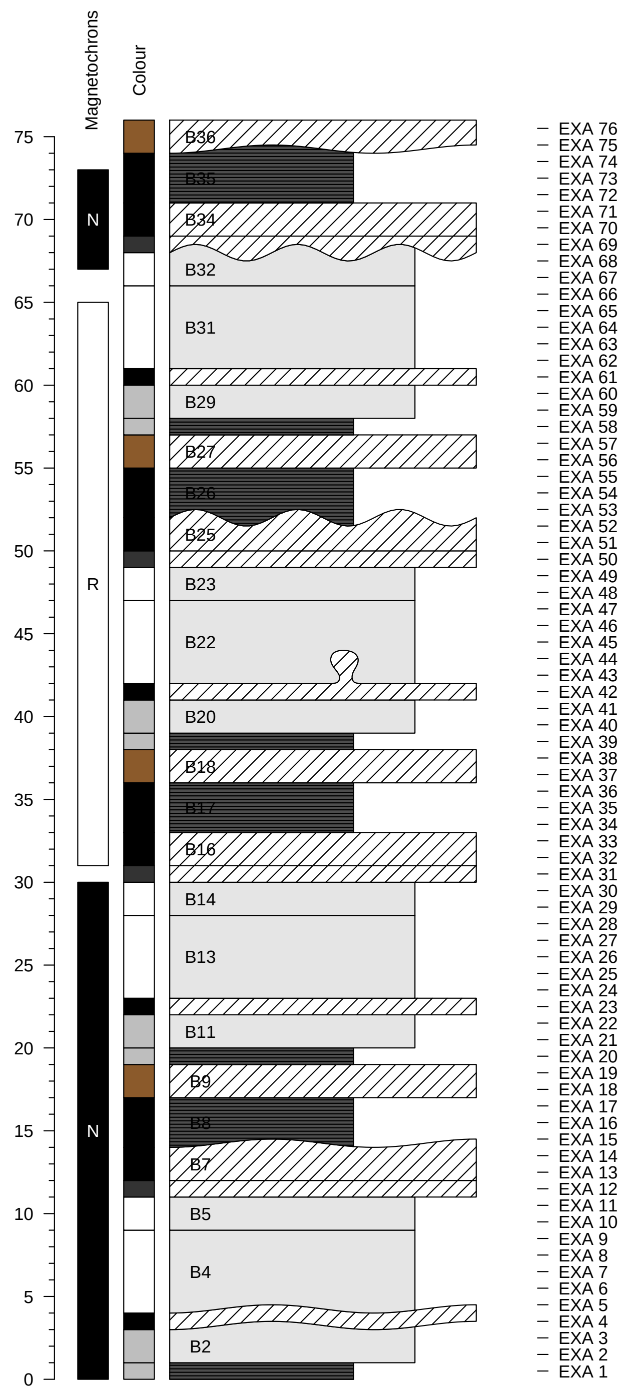


We see that the thickness of beds can vary. Therefore, a bed boundary can actually vary within a given interval. This raises the question of how to document the position of the bed boundaries in data tables that would only have 2 values for the boundaries (lower and upper) rather than 4 (upper and lower interval of variation for the lower boundary, and upper and lower interval of variation for the upper boundary) or even more (detailing the exact form of the boundaries). We propose a convention for the data tables to be used for the generation of lithologs: the positions of the bed boundaries that are defined in the quantified data have to match in a litholog with the positions of the boundaries of the beds on the axis side of the litholog (usually the left side for single logs). The axis side of the litholog is ideal: it follows a straight vertical line, by an implicit convention followed by the large majority of geologists.

Extra stratigraphic or lithological information, such as geomagnetic chrons, rock color, etc., can be added using the `infobar()` function. Any information that can be conveyed by text, such as the positions of samples, can be added using the `axis()` or `text()` functions.

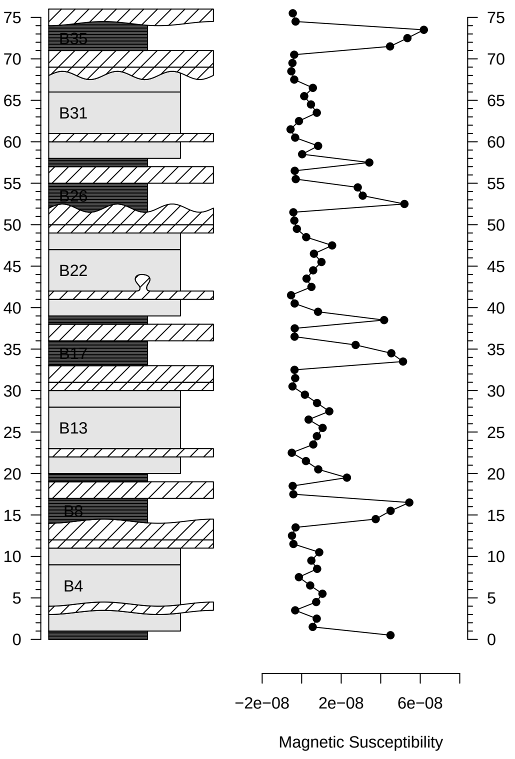
```
# Code repeated from earlier examples ----
basic.log <- litholog(l = bed.example$l, r = bed.example$r,
                      h = bed.example$h, i = bed.example$id)
legend <- data.frame(litho = c("S", "L", "C"), density = c(30, 0,10),
                     col = c("grey30", "grey90", "white"),
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")
s1 <- sinpoint(5,0,0.5,nwave = 1.5)
s2 <- sinpoint(5,0,1,nwave = 3, phase = 0)
s3 <- framesvg(example.liquefaction, 1, 4, 0, 2, plot = FALSE, output = TRUE)
final.log <- weldlog(log = basic.log, dt = boundary.example$dt,
                     seg = list(s1 = s1, s2 = s2, s3 = s3),
                     j = c("s1","s1","s1","s3","s2","s2","s1"), warn = F)
```

```
# Visualizing the resulting litholog (similarly to earlier code) ----
plot.new()
plot.window(xlim = c(-1.5,8), ylim = c(-1,81))
minorAxis(2, at.maj = seq(0, 75, 5), n = 5, las = 1)

multigons(final.log$i, x = final.log$xy, y = final.log$dt,
          col = bed.legend$col,
          density = bed.legend$density,
          angle = bed.legend$angle)

bedtext(labels = bed.example$id, l = bed.example$l, r = bed.example$r,
        x = 0.5, ymin = 2)

# Making a data table for the symbology of magnetochrons
legend.chron <- data.frame(polarity = c("N", "R"),
                           bg.col = c("black", "white"),
                           text.col = c("white", "black"),
                           stringsAsFactors = FALSE)

# Merging symbology with a data table of chrons
chron.legend <- dplyr::left_join(chron.example, legend.chron, by = "polarity")

# Plotting the chrons with the given symbology
infobar(-1.5, -1, chron.legend$l, chron.legend$r,
        labels = chron.legend$polarity,
        m = list(col = chron.legend$bg.col),
        t = list(col = chron.legend$text.col),
        srt = 0)

# Adding color information
colour <- bed.example$colour
colour[colour == "darkgrey"] <- "grey20"
colour[colour == "brown"]    <- "tan4"

# Plotting the color next to the litholog
infobar(-0.25, -0.75, bed.example$l, bed.example$r,
        m = list(col = colour))

text(-0.5, 79, "Colour", srt = 90)
text(-1.25, 79, "Magnetochrons", srt = 90)

axis(4, at = proxy.example$dt, labels = proxy.example$name,
     pos = 6, lwd = 0, lwd.ticks = 1, las = 1)
```



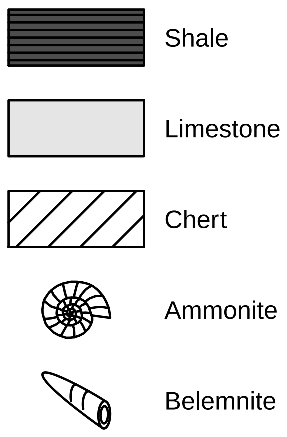
Other plots can be drawn along the litholog. Great care should be taken to ensure that the depth axis is identical in all plots. To ensure that, two components have to be taken into account: the `ylim` argument of `plot.window()` or `plot()` (for vertical logs, otherwise the `xlim` argument), and the graphical parameters defined by the `par()` function, especially the `yaxs` (for vertical logs, otherwise `xaxs`) and the `mar` arguments. The `ylim` argument controls the range of the axis, but the exact range will depend on the `yaxs` argument. Indeed, the default setting of `yaxs` is `"r"`, which stands for regular, and means that the data range defined by `ylim` is extended by 4 percent at each end. Such extension can be unwanted in very long lithologs. Alternatively, the `yaxs` argument can be set as `"i"`, which stands for 'internal', and prevents the extension of the range defined by `ylim`. The `mar` argument controls the margin size of the plotting zone. To add plots along the litholog, a simple way is to use the `mfrow` argument in the `par()` function to define several plotting areas, which will be used by the successively called plots.

```
# Code repeated from earlier examples ----
basic.log <- litholog(l = bed.example$l, r = bed.example$r,
                      h = bed.example$h, i = bed.example$id)
legend <- data.frame(litho = c("S", "L", "C"),
                     col = c("grey30", "grey90", "white"),
                     density = c(30, 0,10),
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")
s1 <- sinpoint(5,0,0.5,nwave = 1.5)
s2 <- sinpoint(5,0,1,nwave = 3, phase = 0)
s3 <- framesvg(example.liquefaction, 1, 4, 0, 2, plot = FALSE, output = TRUE)
final.log <- weldlog(log = basic.log, dt = boundary.example$dt,
                     seg = list(s1 = s1, s2 = s2, s3 = s3),
                     j = c("s1","s1","s1","s3","s2","s2","s1"), warn = F)
# ----

opar <- par() # Save initial graphical parameters (IGP)
par(mfrow = c(1,2),  # Set two vertical plots along each other
    yaxs = "r",                  # Default setting, adds 4% more range for y
    mar = c(5.1, 4.1, 4.1, 0.1)) # Change settings for margins

# Visualizing the resulting litholog (similarly to earlier code) ----
plot.new()
plot.window(xlim = c(0,6), ylim = c(-1,77))
minorAxis(2, at.maj = seq(0, 75, 5), n = 5, las = 1)

multigons(final.log$i, x = final.log$xy, y = final.log$dt,
          col = bed.legend$col,
          density = bed.legend$density,
          angle = bed.legend$angle)

bedtext(labels = bed.example$id, l = bed.example$l, r = bed.example$r,
        x = 0.75, ymin = 3)

# Visualizing quantified values along the litholog ----

par(mar = c(5.1, 0.1, 4.1, 4.1)) # Change settings for margins of 2nd plot

plot.new()
plot.window(xlim = c(-2*10^-8,8*10^-8), ylim = c(-1,77)) # ylim similar to
                                                         # litholog

minorAxis(4, at.maj = seq(0, 75, 5), n = 5, las = 1) # Repetition of the axis to
                                                     # check both sides are matching

lines(proxy.example$ms, proxy.example$dt, type = "o", pch = 19)
axis(1)
title(xlab = "Magnetic Susceptibility")

par(mar = opar$mar, mfrow = opar$mfrow, yaxs = opar$yaxs) # Restore IGP
```



A legend plot can be generated using the `nlegend()` function. The basic idea is to make a subplot for each symbol (using the `par()` function, for instance), in which the `nlegend()` function calls a new plot leaving free space for the symbol (included in [-1, 1], both for x and y coordinates), and adds the text description. This scheme improves automation, e.g., by simplifying the symbol generation of rock types in a function as shown in the code below:

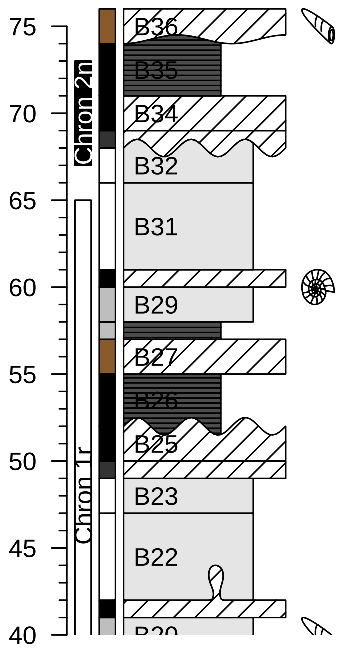
```
legend <- data.frame(litho = c("S", "L", "C"),             # Symbology
                     col = c("grey30", "grey90", "white"), # data table
                     density = c(30, 0,10), angle = c(180, 0, 45),
                     stringsAsFactors = FALSE)

f <- function(legend_row) # To simplify coding, we design here a function
                          # plotting rectangles with the desired symbology
{
   multigons(i = rep(1, 4), c(-1,-1,1,1), c(-1,1,1,-1),
             col = legend$col[legend_row],
             density = legend$density[legend_row],
             angle = legend$angle[legend_row])
}

opar <- par() # Save initial graphical parameters
par(mar = c(0,0,0,0), mfrow = c(5,1)) # Make 5 plot windows

nlegend(t = "Shale", cex = 2) # The cex parameter controls the size of the text
f(1) # 1 stands for the first row of the symbology data table
nlegend(t = "Limestone", cex = 2)
f(2)
nlegend(t = "Chert", cex = 2)
f(3)

nlegend(t = "Ammonite", cex = 2)
centresvg(example.ammonite, 0,0,xfac = 0.5)
nlegend(t = "Belemnite", cex = 2)
centresvg(example.belemnite, 0,0,xfac = 0.5)

par(mar = opar$mar, mfrow = opar$mfrow) # Restore initial graphical parameters
```

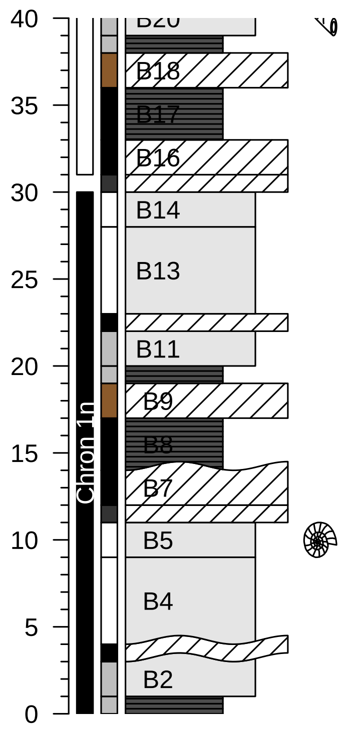


As lithologs can be longer than a single printable page, it is sometimes necessary to split them into separate plots to be displayed on successive pages of a text document. This can be done by grouping all the drawing functions used to generate the litholog into a single function, with `ylim` as an argument. This function can be iterated with successive `ylim` intervals.

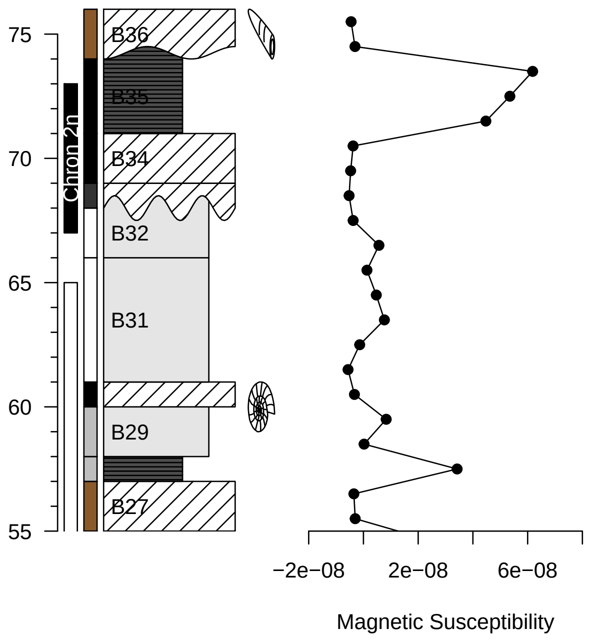
Functions that generate several plots will generate the corresponding pages in the PDF generated by `pdfDisplay()`. All the pages have to be of the same dimensions. To integrate these successive litholog figures into a larger document that would include all the litholog parts, the associated legend, a text description of the section, etc., LaTeX can be used. A \foreach loop in LaTeX can then be applied to import all the pages using the \includegraphics function.

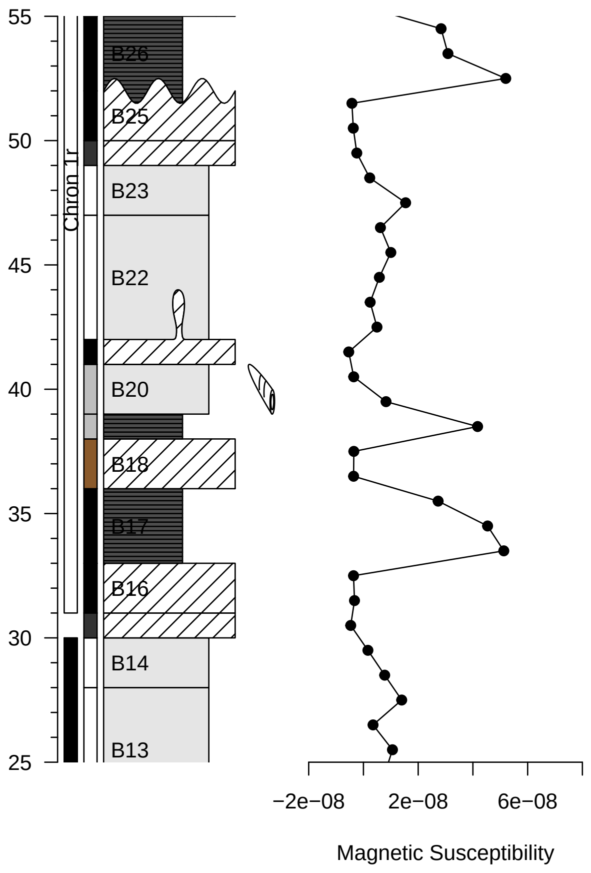
```
# Code repeated from earlier examples ----
basic.log <- litholog(l = bed.example$l, r = bed.example$r,
                      h = bed.example$h, i = bed.example$id)
legend <- data.frame(litho = c("S", "L", "C"),
                     col = c("grey30", "grey90", "white"),
                     density = c(30, 0,10),
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")
s1 <- sinpoint(5,0,0.5,nwave = 1.5)
s2 <- sinpoint(5,0,1,nwave = 3, phase = 0)
s3 <- framesvg(example.liquefaction, 1, 4, 0, 2, plot = FALSE, output = TRUE)
final.log <- weldlog(log = basic.log, dt = boundary.example$dt,
                     seg = list(s1 = s1, s2 = s2, s3 = s3),
                     j = c("s1","s1","s1","s3","s2","s2","s1"), warn = F)
legend.chron <- data.frame(polarity = c("N", "R"),
                           bg.col = c("black", "white"),
                           text.col = c("white", "black"),
                           stringsAsFactors = FALSE)
chron.legend <- dplyr::left_join(chron.example,legend.chron, by = "polarity")
colour <- bed.example$colour
colour[colour == "darkgrey"] <- "grey20"
colour[colour == "brown"]    <- "tan4"
# ----

# Function that will draw a litholog, with personalized coordinates control
log.function <- function(xlim = c(-2.5,7), ylim = c(-1,77))
{
   plot.new()
   plot.window(xlim = xlim, ylim = ylim)
   minorAxis(2, at.maj = seq(0, 75, 5), n = 5, pos = -1.75, las = 1)

   multigons(final.log$i, x = final.log$xy, y = final.log$dt,
             col = bed.legend$col,
             density = bed.legend$density,
             angle = bed.legend$angle)
   bedtext(labels = bed.example$id, l = bed.example$l, r = bed.example$r,
           x = 1, edge = TRUE, ymin = 2)

   centresvg(example.ammonite, 6,
             fossil.example$dt[fossil.example$type == "ammonite"],
             xfac = 0.5)
   centresvg(example.belemnite, 6,
             fossil.example$dt[fossil.example$type == "belemnite"],
             xfac = 0.5)

   infobar(-1.5, -1, chron.legend$l, chron.legend$r,
           labels = chron.legend$id, m = list(col = chron.legend$bg.col),
           t = list(col = chron.legend$text.col))
   infobar(-0.25, -0.75, bed.example$l, bed.example$r,
           m = list(col = colour))
}

# In this gr() function, log.function() is repeated, which plots the
# desired parts of the litholog

gr <- function()
{
   opar <- par()  # Save initial graphical parameters
   par(mar = c(1,2,1,2), yaxs = "i")
   ylim <- c(0,40) # Initial range to be plotted

   for(i in 1:0) log.function(ylim = ylim + 40*i) # Iteration of the plotting
   # The drawing range's length is iteratively added to the range already drawn

   par(mar = opar$mar, yaxs = opar$yaxs) # Restore initial graphical parameters
}

# Integration of gr() in pdfDisplay to make PDFs
pdfDisplay(gr(), name = "divided log", width = 3, height = 5)
```

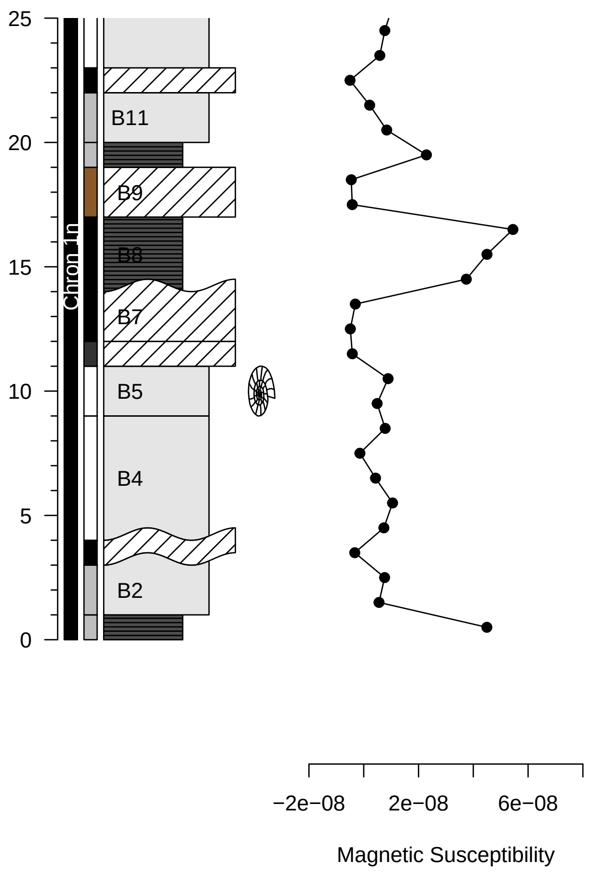




```
# The code can be adapted to divide the plot differently,
# and to add other plots along the litholog

gr2 <- function()
{
   opar <- par() # Save initial graphical parameters (IGP)

   low  <- c(-5, 25, 55) # Another way of defining the dimensions
   high <- c( 25, 55, 85) # of succesive plotting windows

   for(i in 3:1){  # Inverted order to have them in stratigraphic order

      par(mfrow = c(1,2), yaxs = "i") # Plot in two columns, same yaxs for both
      par(mar = c(5,2,1,0)) # Define margins for first plot (left)

      log.function(ylim = c(low[i], high[i]))

      par(mar = c(5,0,1,1)) # Second plot (right): change only the vertical
                            # margins (2nd and 4th)

      plot.new()
      plot.window(xlim = c(-2*10^-8,8*10^-8), ylim = c(low[i], high[i]))
      lines(proxy.example$ms, proxy.example$dt, type = "o", pch = 19)
      axis(1)
      title(xlab = "Magnetic Susceptibility")

   }

    par(mar = opar$mar, yaxs = opar$yaxs, mfrow = opar$mfrow) # Restore IGP
}

pdfDisplay(gr2(), name = "divide in 3", wi = 5, he = 7)
```


75
70
65
60
55
Chron 2n
B36
B35
B34
B32
B31
B29
B27
−2e−08
2e−08
6e−08
Magnetic Susceptibility

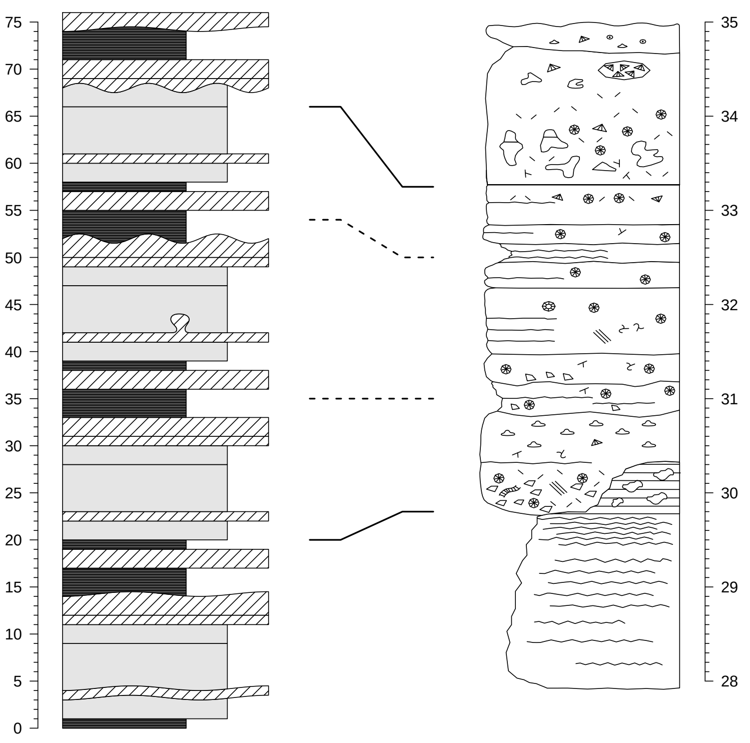

55
50
45
40
35
30
25
Chron 1r
B26
B25
B23
B22
B20
B18
B17
B16
B14
B13
−2e−08
2e−08
6e−08
Magnetic Susceptibility



The preceding examples illustrate some of the capabilities of the **StratigrapheR** package. However, an important question remains unanswered: have we overcome the "from art to useful data" challenge? We will illustrate our answer by importing the computer-drawn litholog from Fig. 1. We will also take the opportunity to show how **StratigrapheR** can help in the comparison and correlation of sections. For that purpose, we plot two lithologs in front of each other and visually link them using the `ylink()` function. `ylink()` currently only works in single window plots, i.e., having a coherent x and y coordinate system. Therefore, we need to change the coordinate system of one of the two lithologs.

Prior to importing it into R using `pointsvg()`, all the lines and polygons in the litholog in Fig. 1 are sparsely interpolated, and all the curves are converted into straight lines. To have perfect positioning in x and y coordinates, the initial drawing is surrounded by a rectangle having known coordinates. Afterward, the figure is saved as an SVG file. All this takes less than a minute with vector graphics software (here using CorelDRAW). The sparse interpolation means that the figures will be angular (take, for instance, the initially elliptical lens containing brachiopods at 34.5 m, when imported by the code here below, it becomes clearly polygonal). If smoother curves are desired, the amount of interpolated points can be increased. When the figure is imported by `pointsvg()`, the rectangle defines the borders of the figure, which by default are set at [-1, 1] in x and y. These coordinates are changed using `framesvg()` by providing the initial coordinates of the rectangle as `xmin`, `xmax`, `ymin`, and `ymax`. Having served its purpose as a reference in x and y, the rectangle can be removed directly in `framesvg()` using the `forget` argument.

```
svg.file.directory <- tempfile(fileext = ".svg")   # Creates temporary file
writeLines(example.HB2000.svg, svg.file.directory) # Writes svg in the file

# Log: 1 Humblet and Boulvain 2000 ----

a   <- pointsvg(svg.file.directory) # Import the svg
out <- framesvg(a,
                xmin = 0, xmax = 5,    # Initial coordinates of the
                ymin = 27, ymax = 36,  # rectangle (see SVG file)
                output = T,  # This allows to output the changed coordinates
                forget = "P287")  # 'forget' removes the rectangle added in the
                                  # svg to serve as a referential in x and y
# Log 2: Code repeated from earlier examples ----

basic.log <- litholog(l = bed.example$l, r = bed.example$r,
                      h = bed.example$h, i = bed.example$id)
legend <- data.frame(litho = c("S", "L", "C"),
                     col = c("grey30", "grey90", "white"),
                     density = c(30, 0,10),
                     angle = c(180, 0, 45), stringsAsFactors = FALSE)
bed.legend <- dplyr::left_join(bed.example,legend, by = "litho")
s1 <- sinpoint(5,0,0.5,nwave = 1.5)
s2 <- sinpoint(5,0,1,nwave = 3, phase = 0)
s3 <- framesvg(example.liquefaction, 1, 4, 0, 2, plot = FALSE, output = TRUE)
final.log <- weldlog(log = basic.log, dt = boundary.example$dt,
                     seg = list(s1 = s1, s2 = s2, s3 = s3),
                     j = c("s1","s1","s1","s3","s2","s2","s1"), warn = F)

# Plotting two logs in front of each other ----

plot.out   <- out # Save a version of the svg object
tie.points <- data.frame(l = c(20,35,54,66),           # Define points to correlate
                         r.raw = c(29.8,31,32.5,33.25)) # the two sections in
                                                        # their own depth scales

plot.out$x   <- 15 - out$x                  # Change the coordinates for
plot.out$y   <- 10*(out$y - 27.5)           # second litholog (imported
axs2         <- 10*(28:35 - 27.5)           # from Fig. 1), to plot it t
tie.points$r <- 10*(tie.points$r.raw - 27.5) # in front of the first litholog

g <- function()
{
   opar <- par()  # Save initial graphical parameters
   par(mar = c(1,4,1,4))
   plot.new()
   plot.window(xlim = c(0,15), ylim = c(0,75))
   minorAxis(2, at.maj = seq(0,75, 5), n = 5, las = 1, cex.axis = 1.2)
   minorAxis(4, at.maj = axs2, labels = 28:35, n = 10, las = 1, cex.axis = 1.2)

   multigons(final.log$i, x = final.log$xy, y = final.log$dt,
             col = bed.legend$col,
             density = bed.legend$density,
             angle = bed.legend$angle)

   placesvg(plot.out, col = "white") # Adding the drawn plot

   ylink(tie.points$l, tie.points$r, 6, 9, ratio = 0.5,  # Correlation between
         l = list(lty = c(1,2,2,1), lwd = 2))            # the two plots

   par(mar = opar$mar) # Restore initial graphical parameters

}

pdfDisplay(g(), "Log Correlation")
```



The most obvious discrepancy between the original computer-drawn version (Fig. 1) and the one imported in R is the lack of color in the latter. We could have identified all the gray polygons one by one and provided them with a color symbology, but such a tedious task would go against the motto of simplifying data management. This highlights that at the moment, the conversion of lithologs "from art to useful data" is not as straightforward as it could be, yet it is not too far out of our reach.

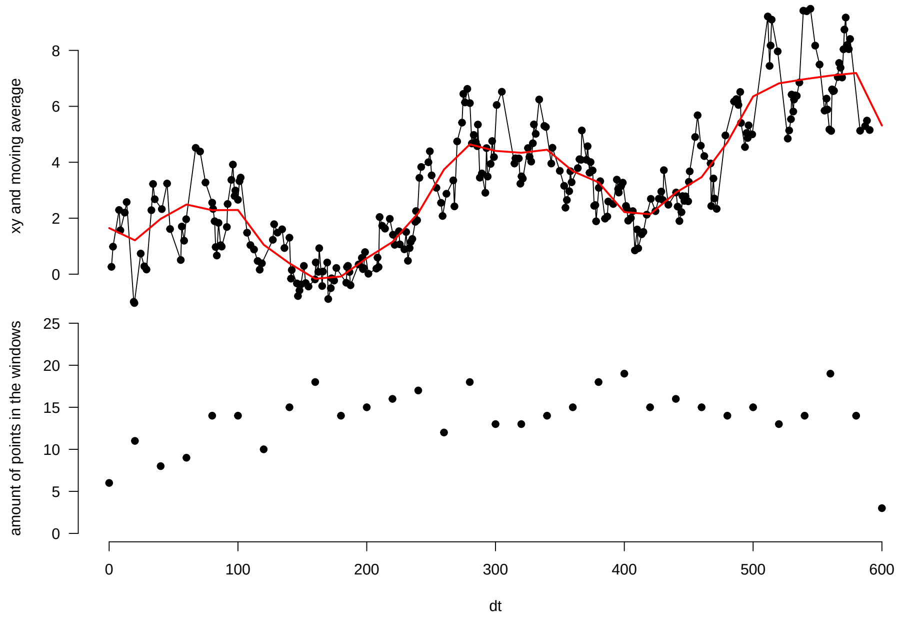
New R functions for geological and general purpose

StratigrapheR was designed in a modular way: low-level general-purpose functions were implemented to simplify the development of higher-level functions. The package also hosts a couple of functions that are not specifically related to lithologs but could be of great use, especially to geologists, and to other developers. We present a few of these functions in this chapter to promote the use of modular and general-purpose functions and to help other developers making their own functions.

- `divisor()`: finds the greatest common rational divisor (GCRD) of a set of values, typically depth, height, or time in time series. This function is important as it allows to transform floating-point values into integers (within the precision range allowed by floating-point arithmetic) by dividing them by the GCRD. We highlight its high potential to automate data processing, especially to interpolate the irregularly-sampled depth, height, or time values that are omnipresent in geology (interpolation by the GCRD preserves the original values). This function is somewhat empirical and would benefit from improvements (among others to reduce the computing time, typically in the case where the GCRD is significantly smaller than the input values), but this would require expertise in mathematics and informatics that the authors do not have. We hope that open-source developers will respond to this challenge.
- `every_nth()`: leaves or removes values at position indexes of multiples of a given amount (n). This is typically useful to discriminate major and minor ticks of a personalized axis.
- `in.window()`: this function can serve as a base for windowing (typically to perform a moving average). It gives a matrix of all the points included in each successive window (in depth, height, or time). We illustrate this with irregularly sampled data points.

```
window <- in.window(irreg.example$dt,  # Depth values
                    w = 30,   # Size of the window
                    xout = seq(0, 600, 20),  # Center position of windows
                    xy = irreg.example$xy)  # Intensity values (or other)
```

```
mov.mean <- rowMeans(window$xy, na.rm = TRUE) # Average of the intensity
                                              # values in windows

presence <- matrix(as.integer(!is.na(window$xy)), # Discriminate between NA
                   ncol = ncol(window$xy))        # values and intensity values
amount   <- rowSums(presence)                     # to determine the amount of
                                                  # real values in each window
                                                  # (example of window calculation)

opar <- par() # Save initial graphical parameters
par(mfrow = c(2,1), mar = c(0,4,0,0))
plot(irreg.example$dt, irreg.example$xy, type = "o", pch = 19,
     xlim = c(0,600), xlab = "dt", ylab = "xy and moving average", axes = F)
lines(window$xout, mov.mean, col = "red", lwd = 2)
axis(2, las = 1)

par(mar = c(5,4,0,0))
plot(window$xout, amount, pch = 19, xlim = c(0,600), ylim = c(0,25),
     xlab = "dt", ylab = "amount of points in the windows", axes = F)
axis(1)
axis(2, las = 1)

par(mar = opar$mar, mfrow = opar$mfrow) # Restore initial graphical parameters
```



- nset(): finds the position of a given amount of values (n) having a common identification code, selecting either the n first or the n last ones, or signaling that they are not available (NA). This is useful to homogenize replicate measurement values.

```
id   <- c("samp1", "samp1", "samp2", "samp3", "samp3", "samp3")
meas <- c(   0.45,    0.55,     5.0,     100,     110,     120)

new_sequence <- nset(id, 2, warn = F)

new_sequence
#>       [,1] [,2]
#> samp1    1    2
#> samp2    3   NA
#> samp3    4    5
```

```
clean_meas <- matrix(meas[new_sequence], ncol = 2)

row.names(clean_meas) <- unique(id)

clean_meas
#>         [,1]   [,2]
#> samp1   0.45   0.55
#> samp2   5.00     NA
#> samp3 100.00 110.00
```

- `seq_mult()`: gives a sequence of numbers that are reordered by a given divisor of the length of the sequence (e.g., `seq_mult(10,5)` gives the sequence 1, 6, 2, 7, 3, 8, 4, 9, 5, 10). This is useful to reorder and manipulate repetitive sequences (e.g., changing 1, 2, 3, 4, 5, 1, 2, 3, 4, 5 into 1, 1, 2, 2, 3, 3, 4, 4, 5, 5 and back).

Present limitations and prospects for the future

StratigrapheR, for the moment, uses the **base** graphics in R. This is a choice that was made in the initial development phase of the package, as the base graphics (also called traditional graphics) are easy to learn for R beginners and are relatively robust, compared to their alternative; the **grid** graphics (Murrell, 2012). The grid graphics are the basis for the **lattice** (Sarkar, 2008) and **ggplot2** (Wickham, 2016) graphical packages. Grid graphics allow more sophistication than the base graphics, but at the price of a more complicated implementation (Murrell, 2012). However, grid graphics would make the entire litholog generation process more efficient, especially by using the concept of grob (which stands for GRaphical OBject). Grobs are R objects that save all the information of a plot, which can then be modified without needing to alter or rewrite the code made to generate the grobs. This would avoid any unnecessary repetition of code. Revisiting the examples in the article, you will see that the code needed for litholog generation does require writing the entire plotting functions at each plot generation or inserting them into a function. On the other hand, elements of a plot made in grid graphics can be expressed via grobs and can be reassembled to generate a modified version of the plot without explicitly making a function or repeating the code. This would further simplify drawing different parts of the same plot on several pages: if a litholog was expressed as a grob, only a few lines of codes would be needed to generate successive versions of the plot, and make them fit on different pages. For all these reasons, grobs would prove to be a key feature for future litholog generation into R. This would especially be useful to integrate lithologs in plots made by other packages, something which was not explored in this article: in the current implementation of **StratigrapheR** (i.e., without grobs), this would be more complicated than it could be (although it should still be possible). We hope to explore this aspect in the subsequent developments of the **StratigrapheR** package.

More generally, the difficulty of importing SVG objects into R should be discussed. With `pointsvg()`, only polyline and polygon objects can be imported; their color, line type, or line thickness are not taken into account. However, this is justified by the fundamental incompatibility between SVG and R graphics, whether from base graphics or grid graphics: SVG files display a wide variety of graphical parameters that are inexistent in R. Other authors have attempted to allow the complete importation of vector graphics into R (e.g., **grImport** (Murrell, 2009), or the **vectoR** package available from GitHub). These works are remarkable but require a lot more effort to use compared to `pointsvg()`. This comes from the fact that the only task allocated to `pointsvg()` is to provide coordinates of polygons and polylines. Afterward, the graphical parameters can be dealt with in R. Therefore we argue that this limitation is not by any means a flaw that will impede the use of **StratigrapheR** or R to deal with geological data. Furthermore, in order to work with `pointsvg()`, one only needs to simplify SVG objects into polygons and polylines. This procedure can be done quite easily in vector graphics software but could also be automated either in R or using SVG-related software and libraries.

Pattern fillings, often used to represent lithologies in traditional lithologs, are currently difficult to plot in R. Indeed, carbonates are often represented with a brick pattern, shales with horizontal layering, and conglomerates by a pattern of polygons to represent heterogeneous pieces. Not all of these pattern fillings are easily implementable in R at the moment, as the only user-friendly pattern is the shading (parallel lines). Rectangular pattern blocs could be generated in SVG form, imported in R using the `pointsvg()` function, and repeated to fill polygons.

Another useful feature that has not been implemented in **StratigrapheR** yet is a way to display 'hardness' variations within the lithological beds. The side opposite to the axis could indeed exhibit continuous variations, which would represent continuous changes in hardness, changes in topographical relief of beds in the field (which is a good indicator of hardness), but also variations of grain size or lithology. These are parameters that are critical to quantify and formalize. We therefore advocate for a community effort to come up with standards for the quantification of the hardness, topographical relief, grain size, and lithology. The way to quantify these parameters should allow to express them in discrete values along the stratigraphical depth or height. These discrete values will make up the points of the polygons symbolizing the beds, on the 'hardness'-varying side of the litholog.

The importation of the PDF documents generated by `pdfDisplay()` back into R, to make documents including the lithologs and providing supporting information (maps, legends, descriptions, etc.), could, in theory, be implemented using the R Markdown scheme (see Xie et al. (2018), Xie et al. (2020) and Allaire et al. (2021) that document the **rmarkdown** package). In practice, however, the `include_graphics()` function in **knitr** (Xie, 2020), which is used to import PDF files in R Markdown, does not allow the selection of specific pages. This means that, for the moment, R Markdown is not well-suited for such a task. Nonetheless, this can be done using LaTeX.

StratigrapheR, for the moment, does not provide a library of geological features symbology, to avoid favoring specific standards of symbology that are not the norm for all geoscientists. However, we encourage the creation of different geological data formats and of their related symbology. One idea would be to have a repository for different geological symbols, grouping different versions of symbols standing for identical geological features and enabling easy download (and upload of new formats).

Finally, the definitive answer to the "from art to usable data" challenge would be to enable the importation into R of lithologs made by other software. This would be easily applicable with ad hoc software tools for which all the geological information is available in a text file. It would furthermore be conceptually possible to import computer-drawn lithologs into Geographical Information System (GIS) software such as the open-source QGIS, treat the polygons and polylines making up the litholog as spatial data, and to couple them with geological meta-data (i.e., by manually selecting these objects and providing them with identification, lithological information, etc.). This could be further facilitated by using algorithms developed for Optical Character Recognition (OCR, typically used to convert handwritten or printed text) and apply them to geological symbols. The combination of polygons, polylines, meta-data, and symbology could subsequently be used as a basis for a general-purpose litholog data format, which could then be imported in R and allow direct figure generation. With this idea, one could make software facilitating the conversion of one geological data format (e.g., hand-drawn lithologs) into this general format and then back to another format (e.g., the LAS format). The final step of this would be to improve and streamline the exchange and publication of geological data.

Summary

StratigrapheR explores new concepts to deal with geological data. It can serve as a strong basis for the generation of lithologs, especially facilitating the workflow when repetitive features are present. The importation of quantified data and the generation of lithologs can be refined to very simple and reproducible steps. Complex drawings can also be included. Modifying the lithologs can be automated, as the geological data can be reprocessed in R or corrected in the files used to generate the lithologs: this means that the visual output can be efficiently updated.

For the future, litholog generation in R has a strong potential to be improved: anyone willing to code in R can put a personal spin on our current work. Ultimately, all types and formats of lithologs could be imported, treated, converted, and exported efficiently, using R as a focal point for geological data processing.

Acknowledgments

The authors would like to thank four anonymous reviewers and the editor Michael Kane, who, through their insight, have brought substantial improvements to the paper and to **StratigrapheR**. We also thank Thomas Goovaerts and an anonymous copy editor for their proofreading of the manuscript. The first author (SW) would like to express his gratitude to Adam Smith for allowing the inclusion of his `every_nth()` function into **StratigrapheR**, and to Michel Crucifix for his help on the `divisor()` function. SW also thanks the Belgian Fund for Scientific Research (FNRS) for the FRIA grant having funded his PhD.

Bibliography

J. J. Allaire, Y. Xie, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, W. Chang, and R. Iannone. rmarkdown: Dynamic Documents for R, 2021. URL https://github.com/rstudio/rmarkdown. R package version 2.7. [p176]

D. W. Bapst. paleotree: Paleontological and Phylogenetic Analyses of Evolution, 2012. URL https://CRAN.R-project.org/package=paleotree. [p154]

D. C. Bowman and J. M. Lees. The Hilbert–Huang Transform: A High Resolution Spectral Method for Nonlinear and Nonstationary Time Series. *Seismological Research Letters*, 84(6):1074–1080, Nov. 2013. ISSN 0895-0695. doi: 10.1785/0220130025. URL https://pubs.geoscienceworld.org/srl/article/84/6/1074/349062/The-Hilbert-Huang-Transform-A-High-Resolution. [p154]

A. Dragulescu and C. Arendt. xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files, Feb. 2020. URL https://CRAN.R-project.org/package=xlsx. [p155]

T. C. Gouhier, A. Grinsted, and V. Simko. R package biwavelet: Conduct Univariate and Bivariate Wavelet Analyses, 2019. URL https://CRAN.R-project.org/package=biwavelet. [p154]

K. Heslop, J. Karst, S. Prensky, and D. Schmitt. Log Ascii Standard (las) Version 3.0. *The Log Analyst*, 40(06), Nov. 1999. ISSN 0024-581X. URL https://www.onepetro.org/journal-paper/SPWLA-1999-v40n6a4. [p154]

M. Humblet and F. Boulvain. Sedimentology of the Bieumont Member: influence of the Lion Member carbonate mounds (Frasnian, Belgium) on their sedimentary environment. *Geologica Belgica*, 2000. ISSN 1374-8505, 2034-1954. URL https://popups.uliege.be/1374-8505/index.php?id=1916. [p153]

S. R. Meyers. Astrochron: An R Package for Astrochronology, 2014. URL https://cran.r-project.org/package=astrochron. [p154]

P. Murrell. Importing Vector Graphics: The grImport Package for R. *Journal of Statistical Software*, 30(4): 1–37, 2009. URL http://www.jstatsoft.org/v30/i04/. [p175]

P. Murrell. *R Graphics*. CRC Press, second edition, 2012. URL https://r-graphics.org/. [p175]

J. R. Ortiz, C. Jaramillo, and C. Moreno. SDAR: Stratigraphic Data Analysis, 2019. URL https://CRAN.R-project.org/package=SDAR. [p154]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. ISBN 978-0-387-75968-5. URL http://lmdvr.r-forge.r-project.org. [p175]

P. Vermeesch. IsoplotR: a free and open toolbox for geochronology. *Geoscience Frontiers*, 9:1479–1493, 2018. URL https://doi.org/10.1016/j.gsf.2018.04.001. [p154]

H. Wickham. *R packages*. O'Reilly, 2015. URL http://r-pkgs.had.co.nz/. [p154]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p175]

H. Wickham, R. François, L. Henry, K. Müller, and RStudio. dplyr: A Grammar of Data Manipulation, Mar. 2020. URL https://CRAN.R-project.org/package=dplyr. [p159]

S. Wouters. DecomposeR: Empirical Mode Decomposition for Cyclostratigraphy, 2020. URL https://CRAN.R-project.org/package=DecomposeR. [p154]

Y. Xie. knitr: A General-Purpose Package for Dynamic Report Generation in R, 2020. URL https://yihui.org/knitr/. [p176]

Y. Xie, J. J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. [p176]

Y. Xie, C. Dervieux, and E. Riederer. *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Raton, Florida, 2020. URL https://bookdown.org/yihui/rmarkdown-cookbook. [p176]

D. Zervas, G. J. Nichols, R. Hall, H. R. Smyth, C. Lüthje, and F. Murtagh. SedLog: A shareware program for drawing graphic logs and log data manipulation. *Computers & Geosciences*, 35(10):2151–2159, 2009. ISSN 0098-3004. doi: 10.1016/j.cageo.2009.02.009. URL http://www.sciencedirect.com/science/article/pii/S0098300409001861. [p154]

Sébastien Wouters
Sedimentary Petrology
University of Liege
Belgium
ORCiD: 0000-0003-2526-0880
sebastien.wouters@doct.uliege.be

Anne-Christine Da Silva
Sedimentary Petrology
University of Liege
Belgium
ORCiD: 0000-0003-4191-7600
ac.dasilva@uliege.be

Frédéric Boulvain
Sedimentary Petrology
University of Liege
Belgium
fboulvain@uliege.be

Xavier Devleeschouwer
O.D. Earth and History of Life
Royal Belgian Institute of Natural Sciences
Belgium
ORCiD: 0000-0002-8841-1159
xdevleeschouwer@naturalsciences.be

dad: an R Package for Visualisation, Classification and Discrimination of Multivariate Groups Modelled by their Densities

by Rachid Boumaza, Pierre Santagostini, Smail Yousfi and Sabine Demotes-Mainard

Abstract Multidimensional scaling (MDS), hierarchical cluster analysis (HCA), and discriminant analysis (DA) are classical techniques which deal with data made of n individuals and p variables. When the individuals are divided into T groups, the R package **dad** associates with each group a multivariate probability density function and then carries out these techniques on the densities, which are estimated by the data under consideration. These techniques are based on distance measures between densities: chi-square, Hellinger, Jeffreys, Jensen-Shannon, and L^p for discrete densities, Hellinger , Jeffreys, L^2, and 2-Wasserstein for Gaussian densities, and L^2 for numeric non-Gaussian densities estimated by the Gaussian kernel method. Practical methods help the user to give meaning to the outputs in the context of MDS and HCA and to look for an optimal prediction in the context of DA based on the one-leave-out misclassification ratio. Some functions for data management or basic statistics calculations on groups are annexed.

Introduction

Techniques such as multidimensional scaling, hierarchical cluster analysis, and discriminant analysis are often used for multivariate data analysis to visualize, organize observations into groups, and model class structure, respectively. These techniques deal with data of the "individuals $\times$ variables" type (Mardia et al., 1979; Krzanowski, 1988), commonly stored in objects of class `data frame`. These techniques are available in the R packages **stats**, **MASS** (Venables and Ripley, 2002), **ade4** (Dray et al., 2007), **FactoMineR** (Lê et al., 2008), **cluster** (Maechler et al., 2019).

In the case where the individuals are organized into occasions or groups, the analyst could be interested in taking into account this data organization by associating with each occasion a mathematical object and performing multivariate techniques on these objects. In the **dad** package (Boumaza et al., 2021), devoted to such data, the objects are probability density functions. These densities are either all continuous (numeric data with Lebesgue measure as reference measure) or all discrete (categorical data with counting measure as reference measure) and are subjected to the following analyses:

- Multidimensional scaling (MDS) of probability density functions aims to visualize a set of densities (or occasions) so that the distances between the densities are preserved as well as possible;
- Hierarchical cluster analysis (HCA) of probability density functions is used to divide a set of densities (or occasions) into clusters so that the densities of the same cluster are as similar as possible and are dissimilar from those of the other clusters;
- Discriminant analysis (DA) of probability density functions deals with the same kind of data, knowing a partition of the densities (or occasions) into classes. Its first objective is to learn how the a priori classes can be explained by the distances between these densities. Then, if the training step is judged satisfactory according to a criterion named misclassification ratio, its second objective is to classify a new density whose class is unknown (Boumaza, 2004).

These three multivariate techniques constitute the core of this work. Theoretically, the **dad** package handles probability density functions and considers multi-group data as samples allowing their estimation. The densities could be considered as functional data processed by packages like **fda** (Ramsay et al., 2020), **fda.usc** (Febrero-Bande and de la Fuente, 2012), or **fdadensity** (Petersen et al., 2019), or as compositional data processed by packages like **compositions** (Tsagris and Athineou, 2020), **Compositional** (van den Boogaart et al., 2020), or **robCompositions** (Filzmoser et al., 2018). The differences or similarities with these approaches are the subjects of part of the discussion of the manuscript (*Discussion and concluding remarks* Section).

These three multivariate techniques are essentially based on distance indices between probability density functions. Literature abounds with such indices: as an example, the encyclopedia of distances of Deza and Deza (2013, p. 235–245) lists some forty. The **dad** package proposes to calculate ten of them among the most common by considering the case of discrete densities and that of continuous densities.

The results returned by the three previous multivariate techniques depend on the distance index used. This is illustrated by simple examples in the context of HCA (Appendix A) or DA (Appendix B), and criteria of distance choice are proposed in the *Practical advice* Section.

Thus, for each distance index, the **dad** package implements:

- its calculation for two densities whose type and parameters are known,
- its estimation for two densities for which there are two samples which allow the estimation of their parameters,
- the generalization of each previous calculation for T ($T > 2$) densities taken two by two, the result of which is a symmetric matrix.

In order to avoid unnecessary redundancies in the entry of data characterizing the groups as these characteristics are the same for all the individuals of the same group, we considered it useful to organize the data in a list of data frames (object of class `folderh`). Also, in order to calculate some statistics (means, covariance matrices...) per group or distances between each pair of groups, we considered it useful to store the individuals of each group in one data frame and these data frames in a list (object of class `folder`). Appendix C details the rationale for introducing these object classes. The functions of the package **dad** implementing the three main techniques (MDS, HCA, DA) apply to such objects. The functions of data management or elementary calculation (means, variance or correlation matrices, moments) applying to these objects could have been made invisible in the package without affecting the presentation of the main techniques. However, to facilitate the work of the analyst interested in processing multi-group data or in experimenting with other multivariate techniques on such data, we have kept them visible. Some of them are presented in Appendix C and Appendix D.

So, the presentation of the **dad** package will begin with a description of the data considered in the previous techniques (*Multi-group data: examples and organization* Section). We will then present functions for the calculation of the distance or divergence measures between discrete densities and between Gaussian densities. The special case of non-Gaussian densities estimated with the Gaussian kernel method is also considered (*Distance / divergence between densities* Section). Then, we will present the functions implementing the three techniques introduced above for the processing of multi-group data: multidimensional scaling (*MDS of densities* Section), hierarchical cluster analysis (*HCA of densities* Section), and discriminant analysis (*DA of densities* Section). Finally, we will give some practical advice (*Practical advice* Section) and briefly highlight some similarities with functions of other R packages (*Discussion and concluding remarks* Section). A summary and appendices complete this presentation.

Multi-group data: examples and organization

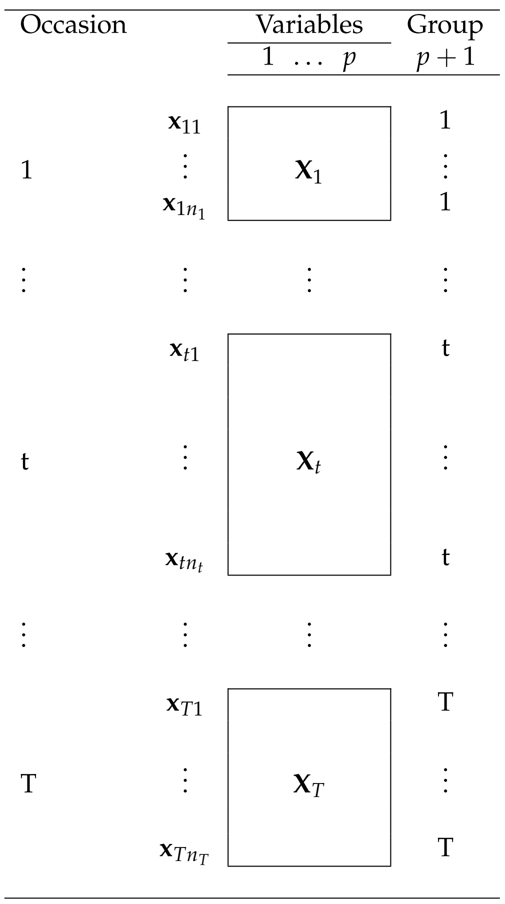
For MDS and HCA, the data $\mathbf{X}$ (Table 1a) of interest have three kinds of objects: occasions $\times$ individuals $\times$ variables. The occasions define a partition of the individuals on which the variables are measured. If T denotes the number of occasions, for each t in $\{1, \ldots, T\}$, the rows of the table $\mathbf{X}_t$ correspond to n_t observations $\mathbf{x}_{t1}, \ldots, \mathbf{x}_{tn_t}$ of X_t a random vector with p components.

For DA, the data of interest are similar to the previous ones with the difference that we have two categories of occasions. The first category consisting of T occasions is partitioned into K subsets deriving from a factor G defined on occasions (Table 2). The second category consists of occasions, numbered $T+1, \ldots$ for which we have data of type $\mathbf{X}$ but not the value of G.

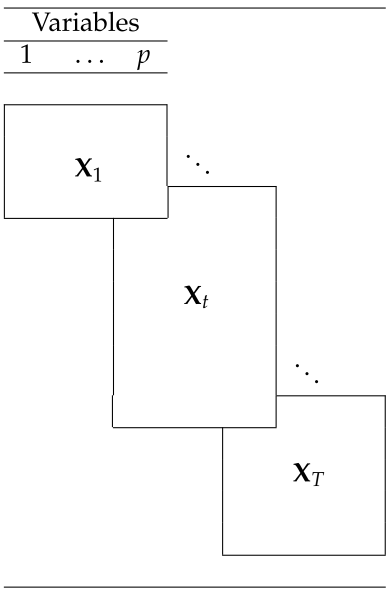
Datasets

The data of the following examples are available in the **dad** package as lists of data frames or arrays, and are loaded by means of the usual `data` function.

1. Archaeological data: the data are stored in `castles.dated`, a list of two data frames whose description is detailed in the subsection *Introductory example* of Appendix C. The data frame `castles.dated$stones` in which for each of $T = 68$ Alsatian castles (occasions), $p = 4$ numerical characteristics are measured on a batch of stones (individuals) used to build the castle. The objective is to:
   - visualize the castles by points in a space of reduced dimension so that the castles having stones of similar dimensions are close to each other and those having stones of very different dimensions are distant;
   - highlight which characteristics of the stones are at the origin of these small or large distances.



(a) Data frame



(b) Data folder

Table 1: For each occasion $t = 1, \ldots, T$, the same p variables are observed for n_t individuals. The data frame (a) consists of $(p+1)$ columns, the last one is a factor designating the occasion. The data folder (b) consists of T data frames each one having the same p column names.

The 68 castles are dated from the period 1140-1650, which is divided into six intervals, numbered 1 to 6, separated by the following cutoff points: 1175, 1210, 1245, 1280, 1350. The building periods are available in the data frame `castles.dated$periods`. The first objective is to analyse the relations between these periods and the previous visualization of the castles. The second objective is to predict the building period of 67 non-dated castles the data of which are in the data frame `castles.nondated$stones`.
This archaeological example (Rudrauf and Boumaza, 2001) was at the origin of the multi-group techniques presented in this work. It illustrates well the MDS and DA techniques even if, as we will see when processing the data, from the archeology point of view, the results are not very satisfactory and have only an indicative value.

2. Agronomic data: the data are stored in the `folderh` object `varietyleaves` consisting of two data frames `variety` and `leaves`, and a key `rose` that is the column name common to the two data frames which connects them. The first data frame `varietyleaves$leaves` is made up of 581 rows (leaves of $T = 31$ rosebushes) and 5 columns corresponding to the number of the rosebush to which the leaf belongs and $p = 4$ numerical characteristics: number of its leaflets, length of its rachis, then the length, and width of its main leaflet. The second data frame `varietyleaves$variety` is made up of T rows and two columns: the number of the rosebush and its variety. There are $K = 6$ varieties (Table 3). This example will illustrate the DA technique. The objective of which is to predict the variety of one plant from measures of several leaves of the plant.

3. Sensory data: the data frame `roses` in which each of $T = 10$ photographs of rosebushes (occasions) was evaluated 3 times, by 14 assessors, for $p = 16$ numerical characteristics, giving a table with 16 columns and 42 rows per rosebush, for a total of 420 rows. In this case, an individual is a couple (assessor, evaluation session). The objective is to visualize the roses and then create clusters of roses that are as similar as possible. A part of these data will illustrate the

Occasion	G
1	1
$\vdots$	$\vdots$
T_1	1
$T_1 + 1$	2
$\vdots$	$\vdots$
$T_1 + T_2$	2
$\vdots$	
$T_1 + \ldots + T_{K-1} + 1$	K
$\vdots$	$\vdots$
$T = T_1 + \ldots + T_{K-1} + T_K$	K
$T + 1$	Not available
$\vdots$	$\vdots$

Table 2: Each occasion t ($t = 1, \ldots, T$) matches a table with n_t rows and p columns (see Table 1). The variable G defined on the occasions takes values $\{1, \ldots, K\}$. For each $k = 1, \ldots, K$, the value k is taken T_k times. The G values of the occasions $T + 1$, ... are not available and have to be predicted.

techniques MDS and HCA.

4. Surveys – census data over years – : from each census conducted in France (INSEE, 2018) during $T = 7$ different years (1968, 1975, 1982, 1990, 2010, and 2015), we extract the population of active individuals aged 25 to 54 and $p = 2$ categorical variables: diploma (4 levels) and socio-professional group (6 levels). From the initial data collected by INSEE, we build a list of $T = 7$ arrays of dimension $(4, 6)$, which is stored in `dspg` object. The objective is to visualize the years and to highlight, when they exist, the temporal evolutions of the frequencies. The MDS technique seems suitable to achieve this objective.
5. Surveys – census data 2015 by department – : from the census of the year 2015 (INSEE, 2018), we consider the same kind of data and organize them in a list of $T = 96$ arrays, which correspond to the 96 departments of metropolitan France. This list is available in the `dspgd2015` object. As for the sensory data, the objective is to visualize the departments and then create clusters of departments that are as similar as possible. In order to give meaning to the clusters, a common and advisable practice is to couple the techniques HCA and MDS.

In the last two examples, the densities are discrete and are given only for illustration. Their detailed presentation is in the **dad** vignette `mds-discrete-distributions`.

Data management

The previously collected data can be organized into:

- A single data frame (Table 1a), by vertically concatenating the T tables $\mathbf{X}_t$, with p columns and appending a factor column designating the occasion, or
- A list of T data frames (Table 1b), each having p columns, as an object from an `S3` class named `folder` (see Appendix C).

Variety	Number of plants	Number of leaves	Number of leaves per plant
Canary	6	65	$= \ 8 + 14 + 11 + 16 + \ 8 + \ 8$
Electron	3	42	$= \ 9 + \ 8 + 25$
Lili Marleen	6	65	$= 15 + 10 + \ 5 + \ 7 + 15 + 13$
Pussta	6	200	$= 35 + 47 + 29 + 20 + 37 + 32$
Starina	5	105	$= 18 + 17 + 19 + 33 + 18$
White Meillandina	5	104	$= 23 + 19 + 16 + 22 + 24$

Table 3: Numbers of plants and leaves per variety and number of leaves per plant.

To carry out discriminant analysis, the training step requires an a priori division of the occasions into clusters, that is, a factor G with K levels defined from the occasion set (Table 2). The predicting step is to assign a level for each occasion whose value of the factor G is not available. The data, therefore, consist of two data frames linked by a hierarchical relationship "1 to N". Each row of the data frame in Table 2 is, thus, matched to several rows of the data frame in Table 1a. The list of these two data frames is an object of `S3` class `folderh` (hierarchical folder) built by the function `folderh` (see Appendix C).

Notice that in the presentation of the data tables, we arranged them so that the individuals (rows of Table 1a) of the same occasion are neighbors and so that the occasions (rows of Table 2) of the same class are neighbors. However, in the **dad** package, such a layout is obviously not necessary. Only the factors $Group$ (Table 1a) and G (Table 2) must be given.

Distance/divergence between densities

In the first subsection, we present the indices which operate on discrete densities. In the second subsection, we consider the indices which operate on Gaussian densities and have an analytical expression depending on the parameters of the densities: means, variances, and covariances. Therefore, these indices can easily be estimated from the parameter estimates. In the third subsection, we present an index based on the estimation of densities on $\mathbb{R}^p$ by the Gaussian kernel method. This index offers the advantage of being easily calculable even for non-Gaussian continuous densities.

The main techniques of the **dad** package depend on the distance index used (Appendix A and B). A brief practical guidance on the choice of distance index is provided in *Practical advice* Section.

Calculation of distances/divergences between discrete densities

Name	Expression
Symmetric chi-square	$\sum_x (p_1(x) - p_2(x))^2/(p_1(x) + p_2(x))$
Hellinger	$\left(2\sum_x (\sqrt{p_1(x)} - \sqrt{p_2(x)}\,)^2\right)^{\frac{1}{2}}$
Jeffreys	$\sum_x (p_1(x) - p_2(x))\ \ln(p_1(x)/p_2(x))$
Jensen-Shannon	$\sum_x (\ p_1(x)\ \ln(2p_1(x)/(p_1(x) + p_2(x))) + p_2(x)\ \ln(2p_2(x)/(p_1(x) + p_2(x)))\ )$
Lp	$\left(\sum_x \lvert p_1(x) - p_2(x)\rvert^p\right)^{\frac{1}{p}}$

Table 4: Distance indices between two discrete densities p_1 and p_2 on the same finite support the states of which are denoted x in the formulas. The sums of the formulas are taken over all the states of the support (Deza and Deza, 2013). The corresponding **dad** functions are: `ddchisqsympar`, `ddhellingerpar`, `ddjeffreyspar`, `ddjensenpar`, and `ddlppar`.

Table 4 lists the expressions of distance indices of two discrete densities and the **dad** functions associated with them. The set of the states of these densities can be either the set of the levels of one categorical variable or the Cartesian product of the q sets of the levels of q categorical variables as in the following example with $q = 2$.

```
> x1 <- data.frame(x = factor(c("A", "A", "A", "B", "B", "B")),
+                  y = factor(c("a", "a", "a", "b", "b", "b")))
> x2 <- data.frame(x = factor(c("A", "A", "A", "B", "B")),
+                  y = factor(c("a", "a", "b", "a", "b")))
> p1 <- table(x1)/nrow(x1)
> p1
   y
x     a   b
```

```
  A 0.5 0.0
  B 0.0 0.5
> p2 <- table(x2)/nrow(x2)
> p2
   y
x     a   b
  A 0.4 0.2
  B 0.2 0.2
```

The $L1$ distance and Jeffreys divergence between the densities $p1$ and $p2$ are computed as follows.

```
> ddlppar(p1, p2)
[1] 0.8
> ddjeffreyspar(p1, p2)
[1] Inf
```

The Jensen-Shannon index is equal to the entropy or average quantity of information of the distribution $(p_1 + p_2)/2$, from which we subtract the sum of the entropies of p_1 and p_2. The other indices are based on the sum of the differences, possibly weighted, between p_1 and p_2 in each state x, unlike the Jensen-Shannon index, which is somewhat more global. These indices are compared in the subsection *Simulated discrete data* of Appendix B.

Calculation of distances/divergences between Gaussian densities

Table 5 lists the parametric expressions of distance indices of multivariate densities and the **dad** functions associated with them. For the univariate case, the expressions are easily deduced.

Name	Expression
Hellinger [(a)]	$\left(2 - 2^{\frac{p}{2}+1} \det(\Sigma V)^{\frac{1}{4}} \det(\Sigma + V)^{-\frac{1}{2}} \exp(-\frac{1}{4}\|\mu - m\|^2_{(\Sigma+V)^{-1}})\right)^{\frac{1}{2}}$
Jeffreys[(b)]	$2^{-1}\,\|\mu - m\|^2_{\Sigma^{-1}+V^{-1}} + 2^{-1}\,\mathrm{tr}((\Sigma - V)(V^{-1} - \Sigma^{-1}))$
L2 [(c)]	$\Big((2\pi)^{-\frac{p}{2}} \det(2\Sigma)^{-\frac{1}{2}} + (2\pi)^{-\frac{p}{2}} \det(2V)^{-\frac{1}{2}} -2\,(2\pi)^{-\frac{p}{2}} \det(\Sigma + V)^{-\frac{1}{2}} \exp(-\frac{1}{2}\|\mu - m\|^2_{(\Sigma+V)^{-1}})\Big)^{\frac{1}{2}}$
L2N [(d)]	$\left(2 - 2^{\frac{p}{2}+1} \det(\Sigma V)^{\frac{1}{4}} \det(\Sigma + V)^{-\frac{1}{2}} \exp(-\frac{1}{2}\|\mu - m\|^2_{(\Sigma+V)^{-1}})\right)^{\frac{1}{2}}$
2-Wasserstein [(e)]	$\left(\|\mu - m\|^2_{I_p} + \mathrm{tr}(\Sigma + V - 2(V^{\frac{1}{2}}\Sigma V^{\frac{1}{2}})^{\frac{1}{2}})\right)^{\frac{1}{2}}$

Table 5: Distance indices between the multivariate Gaussian densities $f \equiv N(\mu, \Sigma)$ and $g \equiv N(m, V)$. (a) The corresponding **dad** function is `hellingerpar`. It is the L^2 distance between the square roots of the densities f and g. (b) The Jeffreys divergence is the symmetrized Kullback-Leibler divergence. Its corresponding function is `jeffreyspar`. (c) The corresponding function of the L^2 distance is `distl2dpar`. (d) L2N, named also normalized L^2 distance, stands for the L^2 distance between $f/\|f\|_{L^2}$ and $g/\|g\|_{L^2}$ and its value is almost similar to the Hellinger distance. Its corresponding function is `distl2dnormpar`. (e) I_p stands for the identity matrix of order p. The corresponding function is `wassersteinpar`.

For example, the Jeffreys divergence is respectively carried out with the `jeffreyspar` or `jeffreys` functions depending on whether the calculations are respectively based on parameters or samples.

```
> m1 <- c(1,1)
> v1 <- matrix(c(4,1,1,9),ncol = 2)
> m2 <- c(0,1)
> v2 <- matrix(c(1,0,0,1),ncol = 2)
> jeffreyspar(m1,v1,m2,v2)
```

```
[1] 5.314286

> library(MASS)
> set.seed(100)
> x1 <- mvrnorm(40, m1, v1)
> x2 <- mvrnorm(30, m2, v2)
> jeffreys(x1, x2)

[1] 6.780999
```

All these indices are based on a combination of the difference between the means μ and m and the dissimilarity between the covariance matrices Σ and V. In the case of equal means, the distance index between Gaussian densities reduces to a dissimilarity between covariance matrices. If the covariance matrices are equal, all these indices reduce to an index of the distance between these means, this index being dependent on the common variance matrix, with the exception of the Wasserstein index, which uses the identity matrix. These indices are also compared in the subsection *Simulated Gaussian data* of Appendix B.

Calculation of L^2 distances between continuous non-Gaussian densities estimated by the kernel method

Except for the L^2 distances, the extension of the other distance indices of Table 5 to any estimated densities still requires a lot of work. Indeed, we experimented with calculating distance indices using numerical integration methods but computation times were so long in the multidimensional case that we did not implement them in the **dad** package. The solution we recommend is to estimate the densities by the Gaussian kernel method and use the L^2 distances.

- **Density estimation with the Gaussian kernel method.** The probability densities f_t are estimated by the Gaussian kernel method:

$$\hat{f}_t(\mathbf{z}) = \frac{1}{n_t|\mathbf{h}_t|^{1/2}} \frac{1}{(2\pi)^{p/2}} \sum_{i=1}^{n_t} \exp(-\frac{1}{2}(\mathbf{z}-\mathbf{x}_{ti})^\top \mathbf{h}_t^{-1}(\mathbf{z}-\mathbf{x}_{ti})), \tag{1}$$

  where $\mathbf{h}_t$ is the non-singular bandwidth matrix and $|\mathbf{h}_t|$ its determinant. This matrix may be provided by the user, or calculated directly according to the AMISE criterion, with reference to the normal distribution (Wand and Jones, 1995), that is:

$$\mathbf{h}_t = h_t \widehat{\mathbf{V}}_t^{1/2}, \tag{2}$$

  with:

$$h_t = \left(\frac{4}{n_t(p+2)}\right)^{\frac{1}{p+4}}. \tag{3}$$

- **Calculation of L^2 distances between estimated non-Gaussian densities.** The calculation of the inner product is carried out with the `l2d` function using a sample per density: `x1` and `x2`; the result derives from the estimation (1) of the densities, the bilinearity of the inner product, and a formula of integral calculus (Wand and Jones, 1995, p. 101). Then, the L^2-distance is directly deduced. This calculation is carried out with the `distl2d` function.

```
> set.seed(40)
> x1 <- c(rnorm(5, mean = 0, sd = 1), rnorm(5, mean = 1, sd  = 2))
> x2 <- c(rnorm(10, mean = 2, sd = 3), rnorm(5, mean = 0, sd = 2))
> distl2d(x1, x2, method = "kern")

[1] 0.2562896
```

The normalized L^2 distance between densities is also possible. Although it has the disadvantage of being time-consuming, it has some similarities to Hellinger's distance, and in the Gaussian case, the two distances have almost the same expression (Table 5).

MDS of densities

Since the function of **dad** package implementing multidimensional scaling of probability density functions is a direct application of the function `cmdscale` of R, it is briefly recalled in the case of continuous densities. The mathematical aspects of this method have been dealt with in several works (Delicado (2011) as MDS; Boumaza (1998), Kneip and Utikal (2001), and Yousfi et al. (2015) as functional PCA). In this section, we will privilege the presentation in MDS form, which offers greater flexibility in the choice of the distance between densities, while taking inspiration from the method of interpretation of the results of PCA developed in Boumaza et al. (2015) in order to interpret the scores resulting from MDS.

Brief presentation of the method

Given T densities and $(\delta_{ts})_{1\leq t,s\leq T}$ the distances/divergences between each pair of them, the MDS technique looks for a representation of the densities by T points in a low dimensional space such that the distances between these points are as similar as possible to the (δ_{ts}) (Cox and Cox, 2001). In R, this multidimensional positioning technique is performed by the `cmdscale` function, whose main argument is the symmetric matrix of distances and the main output is the matrix of coordinates.

If the densities are assumed to be Gaussian, we can use the Hellinger distance, the Jeffreys divergence, the 2-Wasserstein distance, or the L^2 distance (Table 5). If they are not expected to be Gaussian, they are estimated using the Gaussian kernel method, and the only available distance for the moment is the L^2-distance (*Calculation of L^2 distances between continuous non-Gaussian densities estimated by the kernel method* Section).

The **dad** package includes functions for all the calculations required to implement such a method and to interpret its outputs:

- The `fmdsd` function which performs multidimensional scaling and generates scores;
- The `plot` function which generates graphics representing the densities on the factorial axes;
- The `interpret` function which returns other aids to interpretation based on the moments of the variables.

The `fmdsd` function

MDS of densities can be carried using the `fmdsd` function, which applies to an object of the class `folder` (Table 1b). The future or to a data frame and a grouping variable (Table 1a). It is built on the `cmdscale` function of R. In addition to the `add` argument of `cmdscale`, the `fmdsd` function has three sets of optional arguments. The first, consisting of `gaussiand`, `windowh`, and `distance`, controls the method used to estimate the densities and their distances (*Distance/divergence between densities* Section). The second consists of the arguments `data.scaled`, `data.centered`, controls some data transformations, and the logical argument `common.variance`, which, when set to `TRUE`, considers that all the occasions have the same covariance matrix. These three arguments are discussed in Appendix E. The third set consists of optional arguments which control the function outputs.

Interpretation of `fmdsd` outputs

The `fmdsd` function returns an object of `S3` class `fmdsd`, consisting of a list of 11 elements, including the scores, also called principal coordinates, and the moments of the variables per occasion. The outputs are displayed with the `print` function, and graphical representations on the principal planes are generated with the `plot` function.

The interpretation of outputs is based on the relationships between the principal scores and the moments of the densities, in particular their means, variances, covariances, and correlations. These relationships are quantified by correlation coefficients and are represented graphically by plotting the scores against the moments. These interpretation tools are provided by the `interpret` function, which has two optional arguments: `nscores` indicating the indices of the column scores to be interpreted and `moment` whose default value is `"mean"`.

Example

The following example is treated in detail in Boumaza et al. (2015), using PCA of densities. The data consist of $T = 10$ rose bushes assessed three times, by a jury of 14 assessors, for $p = 3$ attributes: top-sided shape (*Sha*), foliage thickness (*Den*), and plant symmetry (*Sym*). Here, we present the results obtained with the MDS technique. This presentation is limited to the major steps in the calculation and the visualization of the results generated by the `fmdsd`, `print`, `plot`, and `interpret` functions.

```
> data("roses")
> rosesf <- as.folder(roses[,c("Sha", "Den", "Sym", "rose")], groups = "rose")
> resultmds <- fmdsd(rosesf, gaussiand = FALSE, distance = "l2")
```

The function fmdsd displays the barplot of the inertia explained by the first nine principal coordinates (Figure 1).

```
> names(resultmds)

 [1] "call"          "group"         "variables"     "d"             "inertia"       "scores"
 [7] "means"         "variances"     "correlations"  "skewness"      "kurtosis"
```

By default, the print function applied to resultmds only displays the names of the variables, the inertia, and the principal coordinates.

```
> print(resultmds)

group variable:  rose
variables:  Sha Den Sym
---------------------------------------------------------------
inertia
  eigenvalue inertia
1    0.02977    25.3
2    0.02261    19.2
3    0.02028    17.2
4    0.01439    12.2
5    0.00980     8.3
6    0.00930     7.9
7    0.00566     4.8
8    0.00344     2.9
9    0.00262     2.2
---------------------------------------------------------------
coordinates
  rose         PC.1         PC.2        PC.3
A    A  0.055191062  0.022167510  0.02655143
B    B -0.004963751 -0.023764758  0.07033084
C    C  0.019611171 -0.122241048 -0.06566866
D    D -0.091777346  0.041132410 -0.04995275
E    E -0.013763431  0.019828288 -0.01341398
F    F  0.016470141 -0.024307858  0.07144865
G    G -0.088949736  0.005722199  0.01223686
H    H -0.025102407 -0.006365474  0.01382440
I    I  0.068203593  0.043999532 -0.02735366
J    J  0.065080705  0.043829197 -0.03800313

> plot(resultmds)
```

The output is shown in Figure 2.

```
> interpret(resultmds)

Pearson correlations between scores and moments
          PC.1  PC.2  PC.3
mean.Sha -0.65  0.29  0.69
mean.Den  0.83 -0.42  0.08
mean.Sym -0.01  0.94 -0.04
Spearman correlations between scores and moments
          PC.1  PC.2  PC.3
mean.Sha -0.65 -0.26  0.65
mean.Den  0.78 -0.30  0.02
mean.Sym  0.16  0.92 -0.35
```

The returned plots of the interpret function are not shown. From the correlations between the principal coordinates (PC) and the means of the variables, we deduce that:

- The higher PC1, the higher "Den", and the lower "Sha" tends to be;
- The higher PC2, the higher "Sym";
- The higher PC3, the higher "Sha" tends to be.

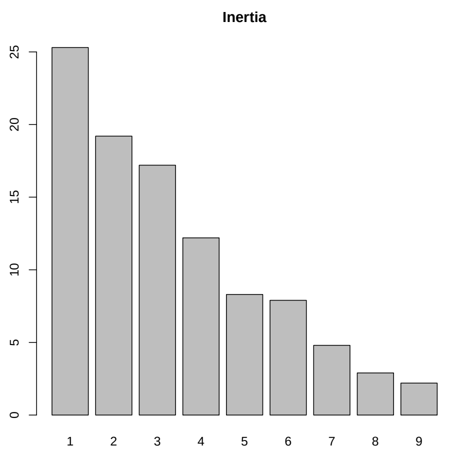


Figure 1: MDS of densities on sensory data (part of the roses data frame): inertia explained by the first ten principal coordinates.

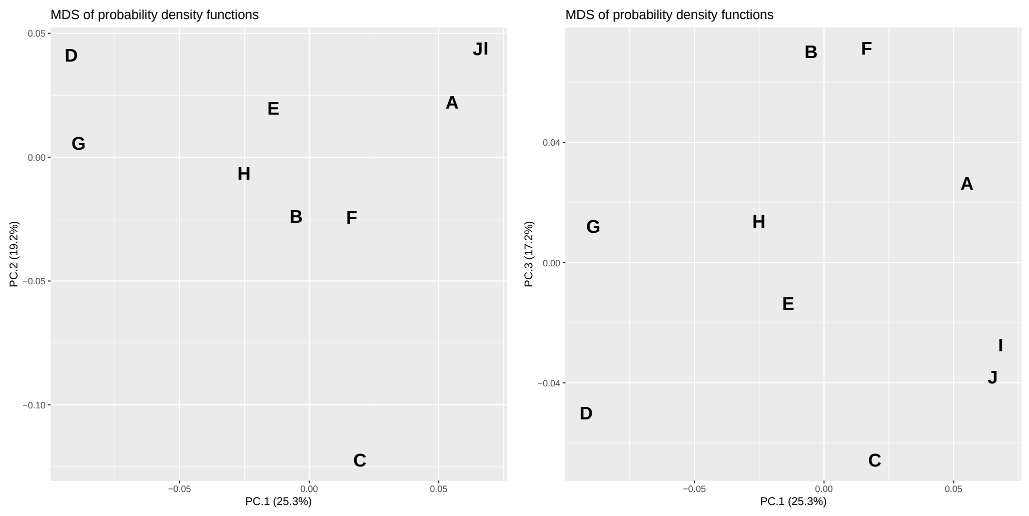


Figure 2: MDS of densities on sensory data (part of roses data frame): the first three principal coordinates.

Thus, from this interpretation of the PCs, we can describe the classes of rose bushes that can be constituted in view of their proximities *vs.* distances visualized in Figure 2. For example, the roses of the class $\{A, I, J\}$ have thick foliage compared to those of the class $\{D, G\}$, the rose bush C is very asymmetrical compared to the other rose bushes, the rose bushes of the class $\{B, F\}$ have a top sided shape.

In order to obtain the correlations between the scores and the standard deviations, we set the optional argument moment to "sd" as in the following example. The other possible values of this argument include "var" (variances), "skewness", "cor" (correlations for multivariate densities).

```
> interpret(resultmds, moment = "sd")

Pearson correlations between scores and moments
        PC.1  PC.2  PC.3
sd.Sha  0.67  0.21 -0.49
sd.Den -0.01  0.77  0.11
sd.Sym -0.13 -0.44  0.76
Spearman correlations between scores and moments
        PC.1  PC.2  PC.3
sd.Sha  0.50  0.45 -0.58
sd.Den  0.15  0.64 -0.08
sd.Sym -0.26 -0.76  0.75
```

Some of the correlations between the PCs and the standard deviations of the variables seem high. Reminding that the PCs are related to means, these correlations are therefore clues of links between standard deviations and means of the variables. We, therefore, represent roses using their means and standard deviations (Figure 3). We highlight that the standard deviations/variances used to assess discordance between assessors tend to be smaller when the products subjected to evaluation on a nine-level scale were awarded marks at the ends of the scale. This result which is actually quite intuitive, obtained by the use of MDS on probability density functions, would have been difficult to demonstrate if the means and standard deviations/variances had been analyzed separately.

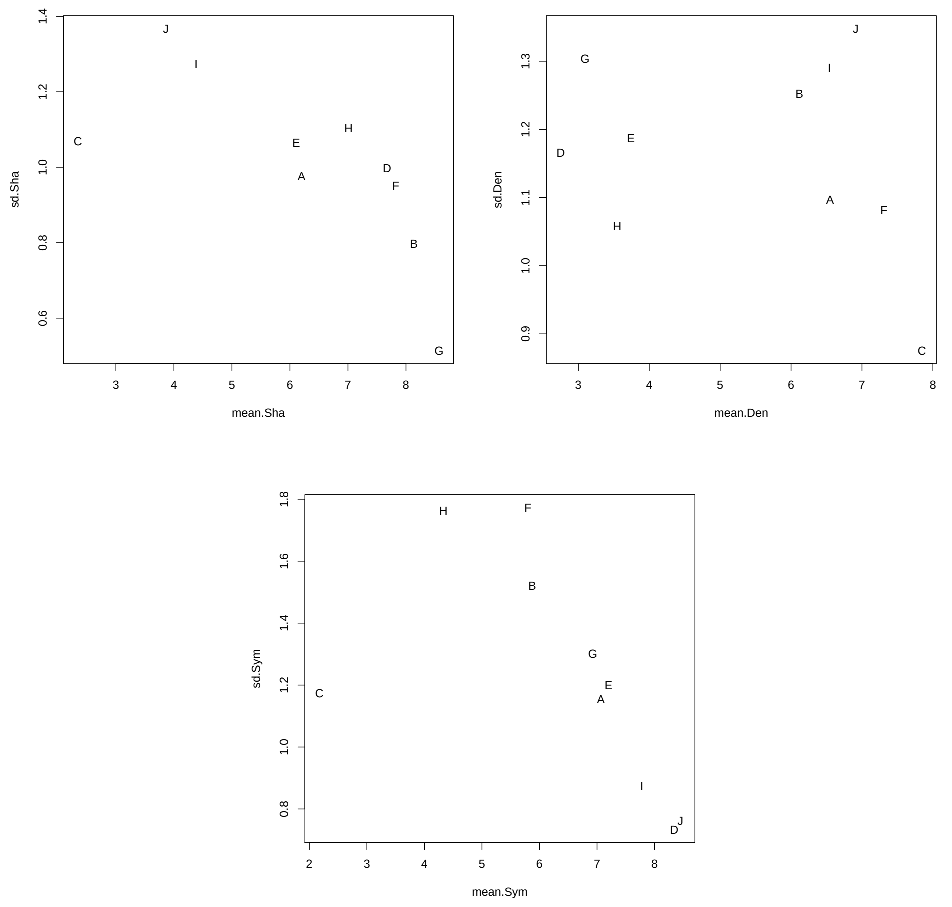


Figure 3: MDS of densities on sensory data (part of `roses` data frame): relationships between means and standard deviations of the variables.

HCA of densities

As for MDS, the `fhclustd` function of the **dad** package implementing hierarchical cluster analysis of probability density functions is a direct application of `hclust`, the corresponding function of R. So it is briefly recalled, and we put the emphasis on the interest of coupling the implementation of MDS with HCA in order to interpret more easily the clusters resulting from HCA.

Brief description of the method and of its outputs

HCA deals with the same kind of data as the MDS technique, namely: T probability density functions and $(\delta_{ts})_{1 \le t,s \le T}$ the distances/divergences between each pair of them. Its purpose is to build a series of nested partitions that can be visualized by means of a dendrogram (Figure 4). An agglomerative building algorithm starts with a partition consisting of T clusters (one density per cluster) then "*it repeats merging the closest pair of clusters according to some similarity criteria until all the data* (densities in our context) *are in one cluster*" (Gan et al., 2007). The criteria are built on dissimilarities between sets of densities, which themselves derive from $(\delta_{ts})_{1 \le t,s \le T}$ obtained as follows:

```
> resulthca <- fhclustd(rosesf, gaussiand = FALSE, distance = "l2")
> round(resulthca$distances, digits = 2)

     A    B    C    D    E    F    G    H    I
B 0.14
C 0.19 0.18
D 0.19 0.18 0.21
E 0.14 0.14 0.18 0.14
F 0.14 0.09 0.18 0.18 0.15
G 0.19 0.15 0.20 0.16 0.17 0.17
H 0.15 0.12 0.17 0.16 0.12 0.14 0.14
I 0.12 0.15 0.18 0.18 0.14 0.15 0.18 0.15
J 0.15 0.16 0.19 0.19 0.15 0.16 0.19 0.15 0.08
```

The dendrogram (Figure 4) is obtained by the following instruction:

```
plot(resulthca, xlab = "Roses", sub = " ", hang = -1)
```

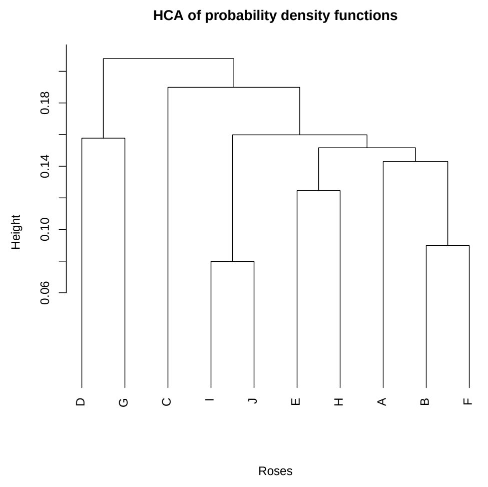


Figure 4: HCA of densities on sensory data (part of roses data frame). The y-axis indicates the distance at which the clusters are merged: the lower this distance is, the more the rosebushes of the clusters are similar.

HCA and MDS on the same densities

By cutting the dendrogram (Figure 4), a partition of the set of the T roses is deduced. For example, in four clusters: $\{D, G\}$, $\{C\}$, $\{I, J\}$, and $\{E, H, A, B, F\}$. This partition is quite similar to the one we could visually make on the basis of Figure 2. This is easy to understand since Figure 4 is only an approximate visualization of the dissimilarities $(\delta_{ts})_{1 \le t,s \le T}$ that served both in MDS and in HCA.

In the *Interpretation of* `fmdsd` *outputs* section, we described the method used to give meaning to the principal coordinates based on the moments of the variables, from which we deduced the meaning of the clusters of roses constituted using the scores of MDS.
This illustrates the process we propose to follow in practice:

- Carry out HCA and deduce a partition in HCA-clusters;

- Carry out MDS and constitute MDS-classes;
- Specify the characteristics of the MDS-classes according to the moments of the variables;
- Describe HCA-clusters which are similar to MDS-classes;
- Describe the other HCA-clusters by using, when possible, the MDS-scores of the groups making up these clusters.

DA of densities

With the notations introduced at the beginning of the section *Multi-group data: examples and organization*, the aim of discriminant analysis of densities (Boumaza, 2004) is to predict the value of G (Table 2) for the occasion $T+1$ represented by the density f_{T+1}, knowing n_{T+1} observations of the random vector X_{T+1}, which are stored in $\mathbf{X}_{T+1}$ (Table 1).

For each $k = 1, \dots, K$, we denote by g_k, the density representing the class k of G. The predicted value is:

$$\widehat{k} = \arg \min_{1 \le k \le K} D(f_{T+1}, g_k), \tag{4}$$

where D is a distance index between densities (*Distance/divergence between densities* section). The data corresponding to the density g_k are those corresponding to the T_k densities f_t belonging to the class k of G. In Appendix F, we specify the possible procedures for calculating the densities g_k and highlight the link between DA of densities and the linear discriminant analysis (**MASS**) in the homoscedastic Gaussian case.

The **dad** package performs all the calculations required to obtain the predicted value through the `fdiscd.predict` function whose first two arguments, `x` and `class.var`, control the input data. It includes the arguments `distance` and `crit`, which respectively set the distance and the densities g_k (Appendix F). It also includes the arguments `gaussiand` and `windowh`, which control the method of density estimation. The `fdiscd.predict` function returns an object of `S3` class `fdiscd.predict`, which is a list consisting of prior and predicted values corresponding to each f_t, a confusion matrix, the distances (d_{tk}) between the f_t's and the g_k's, and proximities. These are calculated from the inverse of the distances in such a way that their sum is 1, but they are not probabilities and are useful for a quick comparison of the distances.

In addition, the package calculates the misclassification ratio for the occasions for which the prior class of G is known through the `fdiscd.misclass` function. This ratio is computed by using the leave-one-out method on the T occasions, and the lower it is, the better the prediction of the variable G by the data $\mathbf{X}$. This function is based on arguments almost identical to those used by the `fdiscd.predict` function. It also generates similar outputs, grouped into an object of the `S3` class `fdiscd.misclass`. However, these two functions differ in the method used to calculate the distances (d_{tk}): if f_t belongs to the k-th class of G, the data corresponding to f_t are not included in the data used to estimate g_k. This function is useful for empirical investigations in order to identify the optimal values of the arguments minimizing the misclassification ratio. These values are then used by the `fdiscd.predict` function for prediction of the G class of an occasion of unknown class.

First example: Castles / Stones

Let us consider the archaeological data (Section *Multi-group data: examples and organization*, Example 1). For the sake of clarity, we combine the six classes to yield three final classes, 1140-1175, 1175-1280, and 1280-1550, numbered 1 to 3.

```
> data("castles.dated")
> levels(castles.dated$periods$period) <- c("1", "2", "2", "2", "3", "3")
> castlesfh <- folderh(castles.dated$periods, "castle", castles.dated$stones)
```

The `fdiscd.misclass` function is used to calculate the misclassification ratios (global and per class).

```
> fdiscd.misclass(castlesfh, class.var = "period", distance = "l2")

misallocation ratio:  0.3382353
            predicted.class
prior.class  1  2  3 total misalloc
          1 10  3  0    13    0.231
          2  5 24  8    37    0.351
          3  0  7 11    18    0.389
```

```
      total 15 34 19    68
--------------------------------------------------------------
 castle prior.class predicted.class misclassed
      1           1               1      FALSE
      2           1               1      FALSE
      3           1               2       TRUE

    ...

    131           2               2      FALSE
    133           3               2       TRUE
    135           2               3       TRUE
    136           2               1       TRUE
```

In order to empirically calibrate the different arguments of the function, the previous operation is repeated, choosing other values for the arguments. If the gaussiand argument is set to TRUE, the smallest global misclassification ratio is 34% (Table 6) and is obtained for `crit = 1`, the default value for this argument, and for `distance = "l2"` or `"hellinger"`.

Estimating the densities by the Gaussian kernel method by setting `gaussiand = FALSE`, with the bandwidth defined by (2) and (3), increased the global misclassification ratio to 40%. If we consider the same proportionality coefficient h in the formula (2), that is

$$\mathbf{h}_t = h\,\widehat{\mathbf{V}}_t^{1/2}, \tag{5}$$

by setting the argument `windowh` of the function `fdiscd.misclass`, we empirically obtain a value of h that minimizes the misclassification ratio. For this purpose, we calculate the misclassification ratio for different values of `windowh` (Table 7). Note that an optimal empirical value is about `windowh = 0.6`, with a misclassification ratio of 32%. One of the best empirical parametrizations of the `fdiscd.predict` function would be `gaussiand = FALSE`, `windowh = 0.6`, and `crit = 1`.

crit	1	2	3
Ratio	0.34	0.56	0.40

Table 6: Castles/Stones. Misclassification ratio assuming densities to be Gaussian and parametrically estimated, depending on the value of the `crit` argument used to select the type of g_k density (Appendix F).

windowh	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Ratio	0.56	0.44	0.38	0.38	0.37	0.32	0.32	0.35	0.34

Table 7: Castles/Stones. Misclassification ratio for densities estimated by the Gaussian kernel method, depending on the value of the `windowh` argument, the proportionality coefficient setting the bandwidth.

The misclassification ratios are disappointing. Thus, the method was used only as an indication to date the castles and its results were cross-checked with other established facts from other historical sources (Rudrauf and Boumaza, 2001).

Second example: Rosebushes/Leaves

Let us consider the agronomic data (Section *Multi-group data: examples and organization*, Example 2). We consider only the continuous variables characterizing the leaves, that is, leaving out the discrete variable “number of leaflets”. As in the first example, we carry out discriminant analysis by applying the `fdiscd.misclass` function for different parameter values. Table 8 gives the classification error rates for the three criteria and some values of the proportionality parameter. One of the best combinations of parameter values would, therefore, be `crit` = 3 and `windowh` = 0.2. The error rate is 0.032, which is clearly better than the result obtained with the archaeological data of the first example. If the same method is applied to all variables, including “number of leaflets”, the error percentage is zero for `crit` = 3 and `windowh` = 0.3.

In these two previous examples, it should be noted that the search for an optimum value of the parameter `windowh` has the disadvantage of being carried out by trial and error and is a time-consuming procedure.

windowh	crit = 1	crit = 2	crit = 3
0.1	0.742	0.806	0.129
0.2	0.452	0.677	**0.032**
0.3	0.226	0.452	0.097
0.4	0.129	0.226	0.097
0.5	0.097	0.129	0.161
0.6	0.097	0.129	0.161
0.7	0.065	0.129	0.194
0.8	0.065	0.097	0.226
0.9	0.065	0.097	0.226

Table 8: Rosebushes/Leaves. Misclassification ratio for the three criteria and different values of the proportionality parameter `windowh`, the densities being estimated by the Gaussian kernel method.

Practical advice

Data management

The functions `folder` and `as.folder` play a central role in the creation of objects handled by the functions `fmdsd` (MDS on continuous data), `mdsdd` (MDS on discrete data), `fhclustd` (HCA on continuous data), and `hclustdd` (HCA on discrete data). The functions `folderh` and `as.folderh` play the same role in DA context: `fdiscd.misclass` and `fdiscd.predict` in the continuous case, or `discdd.misclass` and `discdd.predict` in the discrete case.

It is advisable for the user to know how to use them, especially as their handling is quick and easy (Appendix C).

Computation times

The computation times of the functions of **dad** depend mainly on the computation time of the distances between groups in MDS and HCA contexts or between groups and classes in the DA context. The longest times are achieved when the densities are estimated by the kernel method (*Calculation of L^2 distances between continuous non-Gaussian densities estimated by the kernel method* section). When applying the function `matdisl2` with the option `method = "kern"` on the archeological data (Example 1 of Appendix G), the computation time is approximately five times greater than when `method = "gaussiand"`.

Among the main functions of **dad**, the most time-consuming one is undoubtedly `fdiscd.misclass` when its `gaussiand` argument is set to `FALSE`. That is, the densities are non parametrically estimated. For example, applying this function to archaeological data with `gaussiand = TRUE` takes less than one second while it takes approximately 30-40 seconds with `gaussiand = FALSE` (Example 2 of Appendix G). Thus, we recommend doing tests on small datasets when working on non-Gaussian continuous data.

Choice of distance

The choice of a distance index depends above all on the modeling hypotheses: discrete or continuous data. If they are discrete **dad** proposes five indices (Table 4). If they are continuous, it proposes five indices in the Gaussian case (Table 5) and only one for non-Gaussian data, which is the L^2 distance combined with the estimation of the densities by the Gaussian kernel method.

The distance indices of discrete or Gaussian densities are compared on particular examples in the context of HCA (Appendix A) or DA (Appendix B). It is shown that depending on the technique used and the criterion for measuring the quality of the results, a distance index can be better than another for a set of data and worse for another set. Thus, in the discrete or Gaussian cases, if there is no a priori choice of distance index, we suggest an empirical approach. As the computation times are reasonable, we perform the desired analysis by experimenting with each of the distance indices, then choose the distance index according to one of the two criteria given by **dad**.

For DA, it is the misclassification ratio obtained by the one-leave-out procedure by using the function `fdiscd.misclass`. We should choose the distance index which gives the lowest misclassification ratio.

For MDS, it is the interpretation of principal coordinates in terms of marginal distributions (discrete densities) or moments (continuous densities) using the function `interpret`. We should choose the distance index which gives the highest correlations between principal coordinates and marginal distributions or moments. For HCA, we combine it with the MDS method (*HCA and MDS on the same densities* subsection) and, therefore, we refer to the above to choose the distance index.

Discussion and concluding remarks

The classic statistical methods as multidimensional scaling (MDS), hierarchical clustering analysis (HCA), and discriminant analysis (DA) operate on data which are rows of a data frame. They are available in the R packages **stats**, **MASS** (Venables and Ripley, 2002), **ade4** (Dray et al., 2007), **FactoMineR** (Lê et al., 2008), **cluster** (Maechler et al., 2019). The **dad** package presented in the manuscript generalizes them to data that are organized into groups or occasions. The graphics produced by MDS, the clusters constituted by HCA, or the predicted class provided by DA, concern the occasions and not the individuals who constitute them.

These methods are multivariate data analysis, but they can also be seen as functional data analysis since they operate on multivariate functions estimated from multivariate data. They are theoretically similar to some functional statistical methods implemented in **fda.usc**, **fda**, and **fdadensity** packages. The main difference is in the type of data processed. The functional statistical methods deal with functions of only one or two variables, while the main functions of **dad** can deal with more.

These methods can also be seen as compositional data analysis. Indeed, compositional dataset provides portions of the total (Aitchison, 1986) and the R packages **compositions**, **Compositional**, and **robCompositions** are devoted to such data by implementing many descriptive or graphical techniques and models. Regarding the packages **Compositional** and **robCompositions**, their modeling approach is far enough from our approach to be addressed in this discussion, while the package **compositions** presents similarities with our work which we detail in Appendix H. In the case of a univariate discrete density, the differences concern the method of interpreting the graphical outputs provided by `princomp` of **compositions** and `mdsdd` of **dad** even if we get fairly similar graphics (see the example of Appendix H). In the multivariate case, **dad** takes into account marginal distributions of orders 1 and 2 (or more), while in the current version of **compositions**, only marginal distributions of order 1 are taken into account.

The most recent version of the **dad** package implements functions which operate on discrete data. It also extends the three previous methods to a mixture of numerical and categorical data by transforming the numerical data into categorical data. It is done by dividing the range of each numerical variable into intervals using the function `cut.folder`, an extension of the function `cut` to several variables of a folder.

In addition, the last versions of functions implementing MDS and HCA apply to data stored in a data frame and not only to data stored in a folder.

The following development work is also planned:

- Automation of the empirical search for the proposed proportionality coefficient in the formula (5), considered optimal (in at least some ways) in the context of discriminant analysis;
- Automation of the empirical search for a similar coefficient suitable for use in multidimensional scaling as proposed by Yousfi et al. (2015) in principal component analysis context.

Summary

When we work on multidimensional multi-group data and are interested in the groups and not in the individuals who make up these groups, we want to have statistical methods and computer tools making it possible to describe these groups. The **dad** package is devoted to this.

It mainly provides elaborate functions which implement multivariate techniques such as *multidimensional scaling*, *hierarchical classification analysis*, or *discriminant analysis* on such groups. Moreover, in order to help users in reading the outputs of these techniques, **dad** provides functions for interpreting the results.

It provides datasets which illustrate such data and easy-to-use functions which allow to (i) manage multi-group data by associating a data frame with each group, (ii) compute the distances between groups based on the mathematical concept of probability distribution/density, and (iii) secondarily compute elementary statistics by group as for example frequency distributions and moments.

It defines new data structures called folders (`folder`, `folderh`, `foldermtg`) and provides specific tools to manage them, such as selecting or deleting columns from a `folder`, converting numeric columns of a `folder` to factors. The most noticeable among them allows to easily import into R plant architectures encoded in `mtg` files and, thus, have R packages available to analyze the imported data.

Acknowlegments

The authors thank Gilles Hunault and Julie Bourbeillon, the Associate Editor, and the reviewers for their valuable comments and contributions to this work.

Appendix A: HCA and distance index

We compare three distance indices in the HCA context with a small, simple example originating from an exchange with an anonymous reviewer of a previous version of the manuscript by comparing the $L1$ and $L2$ distances and the symmetrized divergence of Kullback-Leibler. In his review, he made a severe criticism of the $L2$ distance, which was the unique distance proposed in the first versions of the **dad** package: "*Let f a uniform density in the interval* $[0,\ 1]$, *and g also uniform in* $[0,\ 0.90]$. *Clearly, from the point of view of g, it is impossible to reach values in the interval* $(0.90,\ 1]$ *and so, these two densities are not neighbors.*" Indeed, the $L1$ and $L2$ distances and the symmetrized Kullback-Leibler divergence (KL) are:

$$L1(f,g) = 0.20, \;\; L2(f,g) = 0.33, \;\; KL(f,g) = \infty,$$

so $KL(f,g)$ reflects this impossibility of reaching all the values of f from g, unlike $L1$ and $L2$, which consider them to be relatively close to each other.

However, if we add the other two densities v and w, uniform on $[0,\ 0.10]$ and $[0, 0.11]$, and if our objective is to obtain either a partition of f, g, v, and w in 2 classes, or an approximate representation on a plane of these 4 densities, the choice of $L1$ or $L2$ is more informative than KL (Table 9) because with the first two distances, we would easily group f and g on one side and v and w on the other side, whereas with KL all distances are infinite. Our belief is that the choice of a distance between

L1					L2					KL				
	f	g	v	w		f	g	v	w		f	g	v	w
f	0	0.20	1.80	1.78	f	0	0.33	3.00	2.84	f	0	∞	∞	∞
g		0	1.78	1.76	g		0	2.98	2.82	g		0	∞	∞
v			0	0.18	v			0	0.95	v			0	∞
w				0	w				0	w				0

Table 9: $L1$, $L2$, and symmetrized Kullback-Leibler distance/divergence between the uniform probability densities $f = U([0,1])$, $g = U([0,0.9])$, $v = U([0,0.10])$, and $w = U([0,0.11])$.

groups cannot be made in the absolute. It is better to specify a criterion for choosing the distance, which makes sense in the context of the method used, such as percentage of inertia explained in MDS (Delicado (2011)), or in the context of the data analysed. In the *Practical advice* section, we suggest two other criteria.

Appendix B: DA and distance index

In the two examples of *DA of densities* section, the data were not considered plausibly Gaussian and the distance used is the L^2 distance, the only distance computable in **dad** which is suitable for this type of data. In this appendix, we are interested in other types of data and in the misclassification ratios according to the distance index used (*Distance/divergence between densities* section).

Simulated discrete data

Let $p_1 = \mathcal{P}(\lambda_1)$ and $p_2 = \mathcal{P}(\lambda_2)$ be two Poisson distributions with respective parameter $\lambda_1 = 1$ and $\lambda_2 = 2$ and S_1 and S_2 respective simulated samples of size 30. We simulate a sample S of size 10 according to the distribution p_1, we calculate the dissimilarities d_1 and d_2 between this sample and each of the samples S_1 and S_2, then we compare d_1 and d_2. We would expect d_1 to be less than d_2 since the samples S and S_1 are from the same population. Thus, if $d_1 > d_2$ we consider that the sample S is misclassified. We repeat the previous scenario 1000 times and then calculate the misclassification ratio.

The results obtained for the same 1000 samples and each of the distance indices of Table 4 are in column (a) of Table 10. Then we proceed in the same way with 1000 samples of the distribution p_2; the misclassification ratios are in column (b). We notice that for the Jeffreys divergence, the number of

Distance index	(a) Misclassification ratio of 1000 samples from p_1	(b) Misclassification ratio of 1000 samples from p_2
Symmetric chi-square	0.073	0.183
Hellinger	0.064	0.188
Jeffreys	0.049 (600*)	0.139 (643*)
Jensen-Shannon	0.139	0.269
Lp	0.079	0.172

(*) number of samples S at infinite distance from samples S_1 or S_2 .

Table 10: Poisson distributions. Misclassification ratios of 1000 samples from $p_1 = \mathcal{P}_1$ (a) or from $p_2 = \mathcal{P}_2$ (b). The instructions for R allowing the calculation of the ratios corresponding to the symmetric chi-square index are given in the last subsection. The other ratios are obtained by means of the corresponding distance while keeping the same simulation seed.

samples S at infinite distances from S_1 or S_2 is very large, which makes it inefficient at discriminating between samples.
We carried out several simulations as in the previous procedure to compare the distances. We did not find any stability in the order of distances according to the misclassification ratios.

Simulated Gaussian data

As for the discrete case, these distance indices are compared in a simplified context of discriminant analysis using the same procedure. Let p_1 and p_2 be the two Gaussian distributions $N(0,1)$ and $N(1,4)$. The results are given in Table 11. We notice that for the samples from $N(0,1)$ (a), the distance

Distance index	(a) Misclassification ratio for 1000 samples from p_1	(b) Misclassification ratio for 1000 samples from p_2
Hellinger	0.020	0.051
Jeffreys	0.018	0.043
L2	0.034	0.040
L2N	0.026	0.111
2-Wasserstein	0.007	0.169

Table 11: Gaussian distributions. Misclassification ratios of 1000 samples from $p_1 = N(0,1)$ (a) or from $p_2 = N(1,4)$ (b). The R instructions are similar to those of Poisson samples. It suffices to change the function `rpoi` to `rnorm` with the appropriate parameters. In these simulations, the seed is set to 123.

giving the best rate is the Wasserstein distance. On the other hand, it is the worst for the samples from $N(1,4)$ (b).

We carried out several simulations and the conclusion is word for word the same as that of the discrete case with the Poisson distributions.

These results lead us to suggest choosing a distance index based on the minimum misclassification ratio in the subsection *Choice of distance* of *Practical advice* Section.

Computation of the misclassification ratios of Table 10 or 11

The following R instructions are used to compute the misclassification ratios in DA context with samples from Poisson distributions according to the symmetric chi-square distance index of Table 10. The instructions for computing the other ratios of Table 10 and the ratios of Table 11 are given in supplementary material.

```
> n <- 30
> ne <- 10
> nrep <- 1000
> l1 <- 1
> l2 <- 2
> #
> nmisclass1 <- 0
> set.seed(135)
> e1 <- rpois(n, lambda = l1)
> e2 <- rpois(n, lambda = l2)
> for(index in (1:nrep))
+   { x <- rpois(ne, lambda = l1)
+   d1 <- ddchisqsym(x,e1)
+   d2 <- ddchisqsym(x,e2)
+   if(d1 > d2) nmisclass1 <- nmisclass1 + 1
+   }
> misclassratio1 <- nmisclass1 / nrep
> print(misclassratio1)

[1] 0.073

> nmisclass2 <- 0
> set.seed(135)
> e1 <- rpois(n, lambda = l1)
> e2 <- rpois(n, lambda = l2)
> for(index in (1:nrep))
+   { x <- rpois(ne, lambda = l2)
+   d1 <- ddchisqsym(x,e1)
+   d2 <- ddchisqsym(x,e2)
+   if (d1 < d2) nmisclass2 <- nmisclass2 + 1
+   }
> misclassratio2 <- nmisclass2 / nrep
> print(misclassratio2)

[1] 0.183
```

Appendix C: some useful functions on data folders

The **dad** package uses objects of class `folder` or `folderh`. These objects are lists of data frames having particular formats.

Introductory example

Let us consider the archaeological data introduced in *Multi-group data: examples and organization* section. The data that Jean Michel Rudrauf (Rudrauf and Boumaza, 2001) submitted to us was in the form of a paper binder, each sheet of which corresponds to a castle (Figure 5). So for each castle we have:

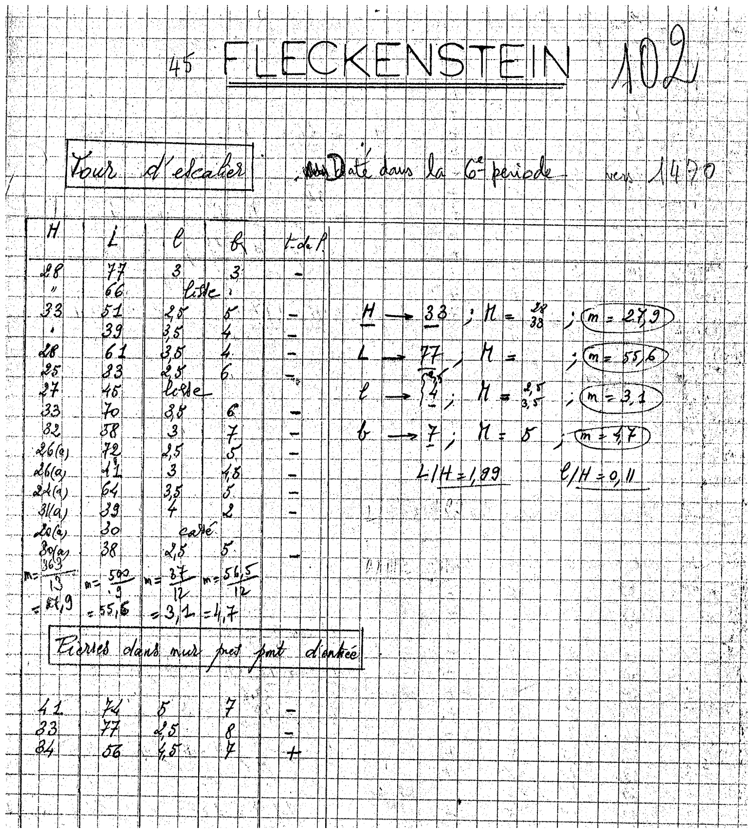
- a name,
- an identification number,
- a building period (sometimes even a year) if it is known,
- the p = 4 measurements of a sample of stones.

A most natural and least redundant way for entering such data is to create:

- one data frame per castle. This data frame with `p` columns is named by the identification number of the castle and contains the measurements of the stones. Its number of rows corresponds to the number of stones whose all measurements are available.
- a data frame whose rows are the castles and whose columns correspond to the name, the identifier and the building period.

From there, we choose to suggest suitable data structures for multi-group data and propose management and calculation tools adapted to these data structures. This is exemplified below.

The archaeological data are stored in the list `castles.dated` of two data frames. The data frame `castles.dated$periods` consists of `T=68` rows (castles) and 2 columns: `castle`, the castle



45 FLECKENSTEIN 102

Tour d'escalier ; Daté dans la 6e periode vers 1470

H	L	l	b	t. de l
28	77	3	3	–
"	66	lisse		
33	51	2,5	5	–
"	39	3,5	4	–
28	61	3,5	4	–
25	83	2,5	6	–
27	45	lisse		
33	70	3,5	6	–
32	58	3	7	–
26(a)	72	2,5	5	–
26(a)	41	3	4,8	–
24(a)	64	3,5	5	–
31(a)	39	4	2	–
20(a)	30	carré		
30(a)	38	2,5	5	–
$m=\frac{363}{13}$ = 27,9	$m=\frac{500}{9}$ = 55,6	$m=\frac{37}{12}$ = 3,1	$m=\frac{56,5}{12}$ = 4,7	

$H \longrightarrow \underline{33}$; $M = \frac{28}{38}$; $\boxed{m = 27,9}$

$L \longrightarrow \underline{77}$; $M =$; $\boxed{m = 55,6}$

$l \longrightarrow \underline{4}$; $M = \frac{2,5}{3,5}$; $\boxed{m = 3,1}$

$b \longrightarrow \underline{7}$; $M = 5$; $\boxed{m = 4,7}$

$\underline{L/H = 1,99}$ $\underline{l/H = 0,11}$

Pierres dans mur près pont d'entrée

41	74	5	7	–
33	77	2,5	8	–
34	56	4,5	7	+

Figure 5: The data patiently collected by Jean Michel Rudrauf corresponds to the castle of Fleckenstein. Its number identification is 102. It was built around 1470 i.e. in the sixth period. The measurements are `H: height`, `L: width`, `l: edging` and `b: boss`. There are 10 stones whose measurements are complete: 7 stones belong to the staircase tower (*Tour d'escalier*) and 3 stones belong to a wall located near the entrance of the castle (*Pierres dans mur près pont d'entrée*). The corner stones designated by the symbol (a) in column `H`, are excluded from the study.

identifier, and `period`, the building period which is a factor with 6 levels. The second data frame `castles.dated$stones` consists of 1262 rows (stones) and 5 columns: 4 numeric characteristics of the stones (`height`, `width`, `edging`, and `boss`) and 1 factor `castle` with 68 levels, which gives the identifier of the castel to which belongs each stone. The implementation of DA requires carrying out calculations not only on all the stones of each castle but also on all the stones of all the castles of each period. To do this, we can store the stones in a data frame made up of 6 columns obtained by adding a column `period` to the file `castles.dated$stones`. Thus, all the stones of a castle have the same `period` value. In order to avoid this redundancy in data entry and management, we propose to store the two files `castles.dated$stones` and `castles.dated$periods` as well as the key `castle` which relates them, in a `folderh` that is a list of two data frames related by a key. The procedure for doing this is given in the paragraph *First example: Castles/Stones* of *DA of densities* section.

Now consider only the `castles.dated$stones` file. To implement both DA and MDS, it is necessary first to calculate and store the vectors of means and the covariance matrices of the `p=4` numeric variables for each castle, then to use the results in the calculation of the distances between each pair of castles. The calculations of means and covariances by castle can be carried out directly from the five-column data frame `castles.dated$stones` using, for example, the following R functions: `colMeans`, `var`, and `by`. In order to facilitate the extraction of data relating to a particular castle and control the data entry, or to check pieces of the computer program, we organize the data in a folder that is a list of `T` four-column data frames (one data frame per castle), and we extend the `mean` and `var` functions so that they apply to such a data structure and return results as lists. So if `x` is the name of the folder, `x[[t]]` is the data frame which contains the data of the castle `t`, `mean(x)[[t]]` is its vector of means and `var(x)[[t]]` is its covariance matrix. We find that by doing so, it is also easier to remember the names and contents of these objects when writing computer programs.

Objects of class `folder`

Such objects, lists of data frames which have the same column names, are created by the `folder` function, from two (or more) data frames, say `x1` and `x2`, as follows: `folder(x1,x2)`.

Optional argument `cols.select` defines the way in which the columns of the two data frames are selected: common columns only (`cols.select = "intersect"`) or all the columns (`cols.select = "union"`). For the `"union"` option, if the data frames do not have exactly the same column names they are complemented by `NA`'s.

The functions `mean.folder` (or simply `mean`), `skewness.folder`, and `kurtosis.folder` applied to the object `x` of the class `folder`, respectively return the list of the vectors of means, skewnesses, and kurtosises of the numeric columns of the elements of `x`. The functions `var.folder` and `cor.folder` return the list of the covariance and correlation matrices. If the data frames of `x` contain non-numeric columns, these functions exclude these columns from the computation. If they contain only one numeric column, these functions return lists of numbers. Note that `map(x,colMeans)`, `map(x,cor)`, and `map(x,var)` return the same values as before, except in the presence of a factor column whose levels are numbers: `map(x,colMeans)` and `map(x,cor)` return an error while `map(x,var)` integrates the factor column in the covariance matrices.

Hence, in `folder` objects, the observed variables are the same on every occasion, unlike individuals which can be different from one occasion to the next. The particular case where individuals are the same on every occasion, corresponds to data defined as three-way data by Kiers (2000) in his essay on standardization of terminology for multiway analysis. In this particular case, it would be better to store the data in an object of class `array`.

Objects of class `folderh`

Such objects are hierarchical lists of data frames in which two successive data frames from the list are related by means of a key. We complete the presentation of the introductory example by a case with more than two data frames. For three data frames, say `df1`, `df2`, and `df3`, there are two keys: the first, say `key1`, describes the "1 to N" relationship between `df1` and `df2`, and the second, say `key2`, describes the "1 to N" relationship between `df2` and `df3`. The arguments of the `folderh` function are introduced in the following order: `df1`, `key1`, `df2`, `key2`, `df3`, and so on, if there are more than three data frames. An example of such object is given in Appendix D.

The function `as.data.frame` applied to such a hierarchical folder, say `fh`, whose constituent elements are listed above, has two main arguments: `key` (the name of a key of `fh`) and `elt` (the name of a data frame of `fh`) with the precision that the value of `elt` is located after the value of `key` in the list of arguments defining `fh`. In the case of two adjacent names, that is `key` = `key1` and `elt` = `df2` or `key` = `key2` and `elt` = `df3`, `as.data.frame` returns a data frame similar to any viewpoint to that returned by the `merge` function. If `key` = `key1` and `elt` = `df3`, the data frame returned by the `as.data.frame` function, say `dfr`, has the same rows as `df3`. The columns of `dfr` are those of the data frames `df3` and `df1`, and those corresponding to all the keys located between `key1` and `df3` in the list defining `fh`, noticing that the key columns are the first columns of `dfr`.

Appendix D: Import plant architectures encoded in `mtg` files

The result of this import procedure consists of an object of class `folderh` and uses an intermediate object of class `foldermtg`. Let us first specify what an `mtg` file is.

The topological structure of a plant is defined from its decomposition into elementary components and the connections between them (Godin and Caraglio, 1998). In Figure 6, the plant is composed of 2 axes (one principal axis, `A1`, coming from the root and bearing one secondary axis, `A2`) and each axis is composed of internodes and peduncles: the principal axis is composed of seven internodes `I1`, ..., `I7` and one peduncle `F1` and the secondary axis is composed of three internodes `I8`, `I9` and `I10`. Among the computer file types used to store the topology of a plant, we are interested in `mtg` (multiscale tree graph) files which can be opened with a spreadsheet as LibreOffice-Calc, or Excel (Pradal and Cokelaer, 2010). Breaking a plant into axes and breaking each axis into internodes create two "1 to N" relationships which can be stored in an object of class `folderh`. This hierarchical folder, say `fh`, is a list of three data frames `P` (set of 1 plant), `A` (set of 2 axes), and `I` (set of 10 internodes), and two keys `P` and `A`. It is the result of the following three R instructions which successively imports an mtg file into R, creates an object of the `S3` class `foldermtg`, and then creates `fh`.

```
> mtgfile <- system.file("extdata/plant2.mtg", package = "dad")
> x2 <- read.mtg(mtgfile)
> fh <- as.folderh(x2, classes = c("P", "A", "I"))
> print(fh)

$P
      P Variety
v01 v01 Starina

$A
      P   A Length
v02 v01 v02     30
v07 v01 v07     12

$I
      A   I Leaflet
v03 v02 v03       3
v04 v02 v04       3
v05 v02 v05       5
v06 v02 v06       5
v08 v07 v08       5
v09 v07 v09       5
v10 v07 v10       7
v11 v02 v11       7
v12 v02 v12       5
v13 v02 v13       3

attr(,"class")
[1] "folderh"
attr(,"keys")
"P" "A"
```

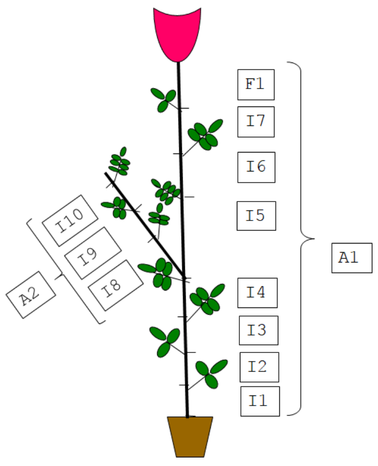


Figure 6: Decomposition of a plant (P) into axes (A), internodes (I), and peduncles (F).

Biological component		File code
/P1		v01
∧/A1		v02
∧/I1		v03
∧<I2		v04
∧<I3		v05
∧<I4		v06
	+A2	v07
	∧/I8	v08
	∧<I9	v09
	∧<I10	v10
∧<I5		v11
∧<I6		v12
∧<I7		v13
∧<F1		v14

Table 12: The first two columns correspond to the plant topology stored in an `mtg` file. The third column corresponds to the code of each component: the plant (P) is encrypted `v01`, etc.

An object of class `foldermtg` is a list of data frames. It is only an intermediary to which the only function of R which can be applied to it is `as.folderh`. This, therefore, makes it possible to retrieve a hierarchical folder which consists of data frames, one data frame per type of biological components, on which one can operate statistical calculations by means of R functions.

Appendix E: Particular cases of the `fmdsd` function

We assume that the densities associated with the occasions are Gaussian. For particular values of the arguments `data.centered`, `data.scaled`, and `common.variance`, the outputs of the fmdsd function are similar to those returned by other functions.

`common.variance = TRUE`

The Gaussian densities are assumed with the same covariance matrix which is estimated using all data. Thus, the distances between the densities are reduced to the differences between their mean vectors (Table 5). The Euclidian distances between mean vectors computed by the R function `dist` are equal to that computed by the 2-Wasserstein distance between densities. In this case, the R function `cmdscale` applied on the mean vectors returns exactly the outputs of the `fmdsd` function.

`data.centered = TRUE`

The Gaussian densities are assumed with zero mean vectors. Consequently, the distances between the densities are reduced to the differences between their covariance matrices (Table 5). With the 2-Wasserstein distance, the distances between the densities are reduced to the Hilbert-Schmidt distances between the square root of the covariance matrices while the dual STATIS method (Lavit et al., 1994) uses Hilbert-Schmidt distances between the covariance matrices. In R, the calculations can partly be performed with the `DSTATIS` function of the **multigroup** package (Eslami et al., 2013, 2020) or the `statis` function of the **ade4** package (Dray et al., 2007) after some transformation. However, the outputs of these two functions are completely different from those of the `fmdsd` function (*MDS of densities* Section): `fmdsd` focuses on the visualization of the occasions, while the other functions focus more on the visualization of variables or individuals.

`data.scaled = TRUE`

With the two previous special cases, we have shown that MDS on densities is a way to take into account globally the means, variances, and covariances of the occasions and, therefore, a form of generalization of separate analyses either on averages or on variances and covariances. The optional argument `data.scaled` is useful when the analyst is interested to focus only on the relationships between variables.

Appendix F: DA of densities and classic DA

Densities associated to the classes of the factor G defined on the occasions

In the functions `fdiscd.misclass` and `fdiscd.predict`, for each $k = 1, \ldots, K$, the density g_k representing the class k of G is estimated using a procedure selected by means of the argument `crit`. Recalling that the class k contains T_k densities f_t, the three procedures are defined as follows.

1. All samples related to the T_k occasions of class k are pooled and constitute a single sample, which is then used to estimate g_k.
2. If $\hat{f}_t$ estimates f_t, then g_k is estimated by the mean value of the T_k densities pertaining to the class k: $(1/T_k)\sum \hat{f}_t$.
3. The mean value of the previous T_k densities is calculated by weighting each $\hat{f}_t$ by the size of its corresponding sample: $(1/\sum n_t)\ \sum n_t \hat{f}_t$.

The last two procedures are only available if the argument `distance` is set to `"l2"`. If there is only one occasion per class that is $T_k = 1$, $\forall k$, the three procedures are the same. In this case, the data are those of the training step of classic DA. However, in the prediction step, we have to assign a group of individuals not individually but taken as a whole.

Homoscedatic Gaussian case

We assume that:

- the densities f_t $(t = 1, \ldots, T)$ are Gaussian with the same covariance matrix V. We denote f_t by $N(m_t, V)$;
- there is one density f_t per class of G. So, there are $K = T$ classes, and the density of the class k is denoted by f_k.
- the density f_{T+1} is Gaussian $N(m_{T+1}, V)$

We have to predict the class value of f_{T+1} using the rule 4 and one of the distances from Table 5. For `distance = "wasserstein"`, we calculate the distances between the mean vectors $\|m_{T+1} - m_k\|_{I_p}$ $(k =$

$1,\ldots,T)$, and for the other distances we calculate the distances $\|m_{T+1} - m_k\|_{V^{-1}}$ $(k = 1,\ldots,T)$.
In the homoscedastic Gaussian case, this makes DA of densities appear as a form of extension of linear discriminant analysis where the group of individuals to be predicted is summarized by the mean vector of the group.

Appendix G: Some computation times

All the following calculations are carried out using a laptop computer equipped with an i5 processor on the archaeological data. The stone characteristics are stored in the data frame `x.df`. Then, we create the corresponding folder `x.folder`.

```
> data(castles.dated)
> x.df <- castles.dated$stones
> x.folder <- as.folder(x.df, groups = "castle")
```

Example 1: Comparison of distance indices in the continuous case

The computation times of the inter-group distances are of the same magnitude when the densities are supposed Gaussian and parametrically estimated. Computations with the L^2 distance take about one second.

```
> system.time(matdistl2d(x.folder, method = "gaussiand"))

       user      system     elapsed
       1.13        0.06        1.36
```

When the densities are estimated by the kernel method, the computation time is multiplied by about 5.

```
> system.time(matdistl2d(x.folder, method = "kern"))

       user      system     elapsed
       6.38        0.14        6.53
```

Example 2: Computation time of the function `fdiscd.misclass`

We build the hierarchical folder `x.fh` corresponding to the archeological data. We first apply the function `fdiscd.misclass` with the option `gaussiand = TRUE` then with the option `gaussiand = FALSE`.

```
> x.fh <- folderh(castles.dated$periods, "castle", castles.dated$stones)
> system.time(fdiscd.misclass(x.fh, class.var = "period", distance = "l2", gaussiand = TRUE))

       user      system     elapsed
       0.16        0.02        0.17

> system.time(fdiscd.misclass(xfh, class.var = "period", distance = "l2", gaussiand = FALSE))

       user      system     elapsed
      38.13        0.17       38.64
```

Appendix H: MDS on compositional data and compositional data analysis

The discrete densities considered in **dad** are compositional data as they are made up of positive numbers whose sums are one. It is possible to apply to them the techniques developed in the package **compositions** after an adapted data formatting work. The opposite, i.e., applying **dad** techniques to compositional data, is partly true as in the following example illustrating the functions `princomp.acomp` of **compositions** and `mdsdd` of **dad** on the same data.

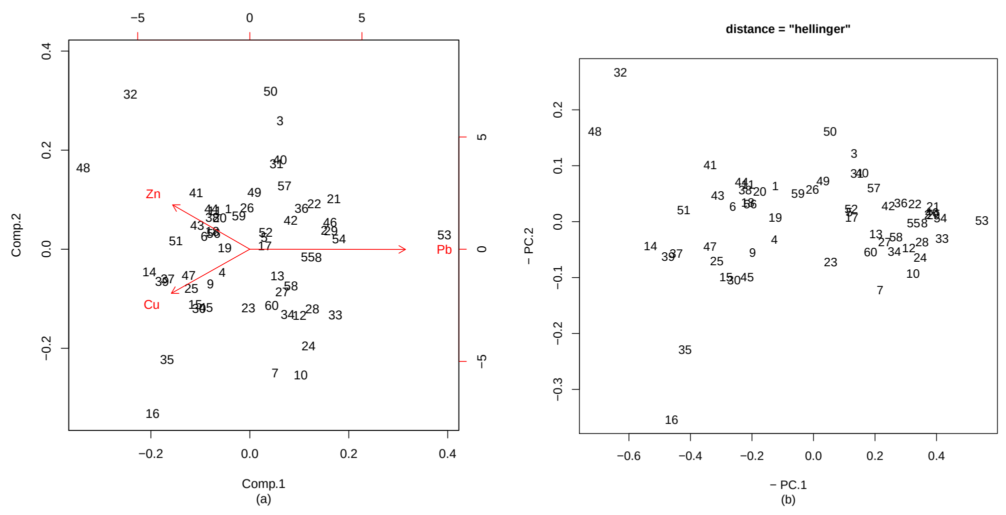


Figure 7: Compositional data. (a) The two first principal components of `princomp.acomp` applied on `sa.lognormals` data. (b) The two first principal coordinates of `mdsdd` on the same data. The signs of the scores are chosen so that the comparison of the two graphs is easier.

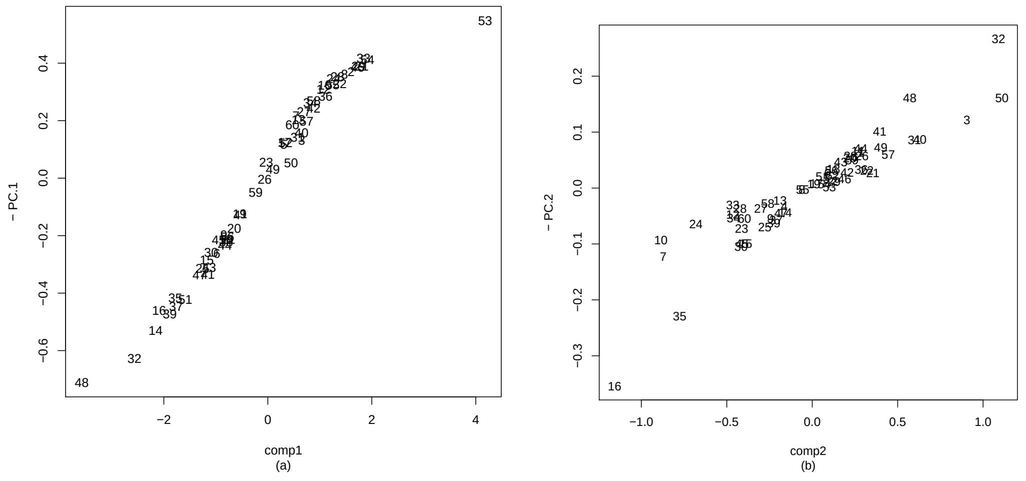


Figure 8: Plots of the scores of the figure 7a and those of the figure 7b. The correlation between `comp1` and `-PC.1` is 0.98 and between `comp2` and `-PC.2` is 0.93.

Data loading

The simulated data `sa.lognormals` of **compositions** are stored in a matrix 60 × 3 whose columns are the amounts of `Cu`, `Zn`, and `Pb` present in 60 samples. They are loaded as follows.

```
> library(compositions)
> data(SimulatedAmounts)
> print(sa.lognormals)
```

```
              Cu         Zn         Pb
 [1,]  8.8043262 35.1671810  45.895025
...
[60,]  3.9854998   6.1301909  40.579417
```

These initial data are transformed so that each row is of sum 1 by means of the acomp function.

```
> acomp(sa.lognormals) -> x1
> print(x1)
```

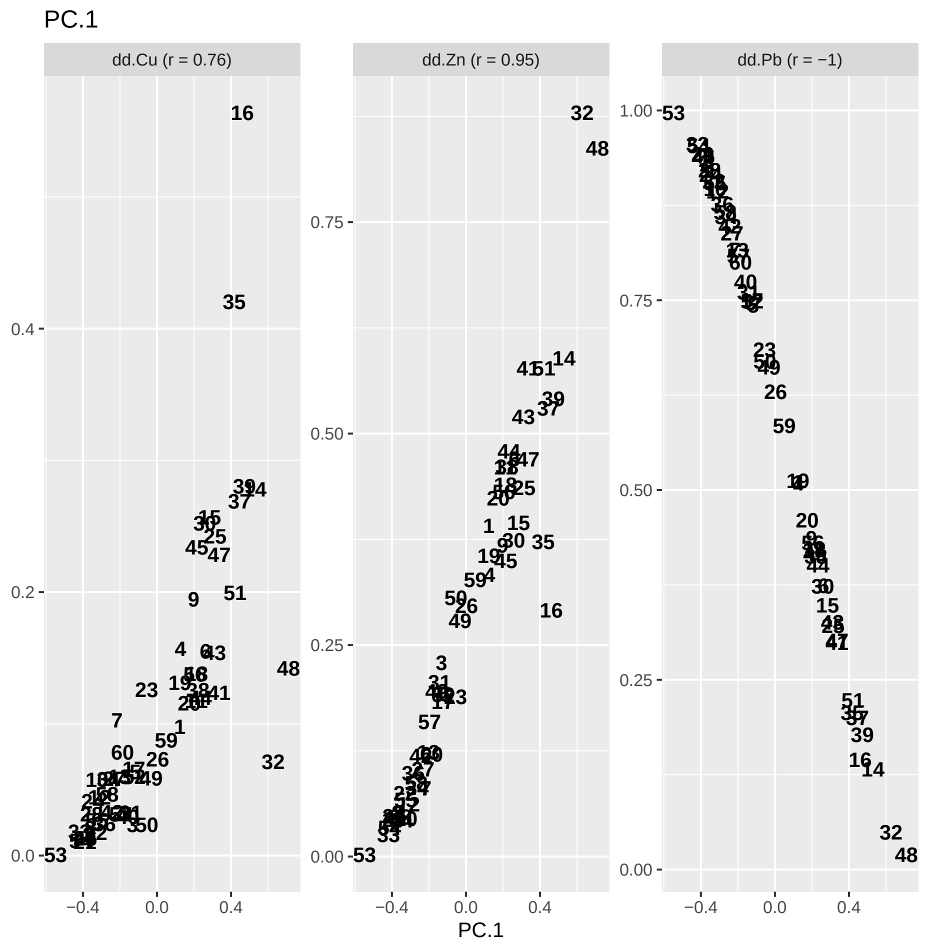


Figure 9: MDS of compositional data considered as discrete densities. Plotting `PC.1` against the relative amounts of `Cu`, `Zn`, and `Pb`.

```
      Cu          Zn          Pb
 [1,] 0.097971136 0.391326782 0.51070208
...
[60,] 0.078617049 0.120922730 0.80046022
attr(,"class")
[1] acomp
```

The rows of the object `x1` of class `acomp` (relative amounts of `Cu`, `Zn`, and `Pb` present in 60 samples) are transformed into tables or arrays. These tables or arrays then are organized in a list to be subjected to `mdsdd`. The three levels of the unique categorical variable are denoted `dd.Cu`, `dd.Zn`, and `dd.Pb`.

```
> x <- as.data.frame(x1)
> nomscol <- colnames(x)
> x2 <- list()
> for(i in 1:60) {x2[[i]] <- as.table(as.numeric(x[i,]));
+  dimnames(x2[[i]]) <- list("dd" = nomscol)}
> names(x2) <- rownames(x)
```

Outputs of `princomp` and `mdsdd`

By applying the `princomp.acomp` (or simply `princomp`) function to `x1`, we obtain the visualization of the data in a biplot (Fig. 7a).

The `mdsdd` function (MDS of discrete densities) is applied to `x2` which are the rows of `x1` considered as discrete densities. We choose to present the results for the `distance` argument set to `hellinger`. These results are quite similar to those obtained with other distances as `chisqsym`, `jeffreys`, or `jensen`.

```
r2 = mdsdd(x2, distance = "hellinger")
```

It provides the figure 7b.

The two figures are almost equivalent: by plotting the coordinates `comp1` and `comp2` against `PC.1` and `PC.2` of the two previous graphics we get the figure 8.

Interpretation of the scores

For the interpretation of the axes, the two packages provide quite different tools. With **compositions**, the biplot makes it possible to visualize the links between the variables `Cu`, `Zn`, and `Pb` and their links with the principal components. With **dad**, the interpretation is done by crossing the scores and the initial data. The interpretation is based on the strength of the links between the PCs and the probabilities of occurence of each level. In the compositional data example, almost a single axis would suffice to explain the general trend: the first principal coordinate `PC.1` explains 90% of the inertia.

```
> interpret(r2, nscore = 1)

Pearson correlations between scores and probability distributions of each variable
...
Spearman correlations between scores and probability distributions of each variable
       PC.1
dd.Cu  0.90
dd.Zn  0.96
dd.Pb -1.00
```

The graphical output is shown in Figure 9. The relative amounts are highly correlated with `PC.1`, showing that a low relative amount of the `dd.Pb` level corresponds to a high relative amount of the levels `dd.Cu` or `dd.Zn`. It is the result suggested by the biplot (Fig. 7a) returned by the function `princomp.acomp` of the package **compositions**.

Bibliography

J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman & Hall, London, 1986. URL http://links.jstor.org/sici?sici=0035-9246%281982%2944%3A2%3C139%3ATSAOCD%3E2.0.CO%3B2-9. [p194]

R. Boumaza. Analyse en composantes principales de distributions gaussiennes multidimensionnelles. *Revue de Statistique Appliquée*, XLVI(2):5–20, 1998. URL http://www.numdam.org/item?id=RSA_1998__46_2_5_0. [p186]

R. Boumaza. Discriminant analysis with independently repeated multivariate measurements: An l^2 approach. *Computational Statistics and Data Analysis*, 47(4):823–843, 2004. URL https://doi.org/10.1016/j.csda.2004.01.001. [p179, 191]

R. Boumaza, S. Yousfi, and S. Demotes-Mainard. Interpreting the principal component analysis of multivariate density functions. *Communications in Statistics - Theory and Methods*, 44(16):3321–3339, 2015. URL https://doi.org/10.1080/03610926.2013.824103. [p186]

R. Boumaza, P. Santagostini, S. Yousfi, G. Hunault, J. Bourbeillon, B. Pumo, and S. Demotes-Mainard. **dad**: *Three-Way / Multigroup Data Analysis Through Densities*, 2021. URL https://CRAN.R-project.org/package=dad. R Package version 3.4.6. [p179]

T. Cox and M. Cox. *Multidimensional scaling*. Chapman & Hall/CRC, Boca Raton, 2001. URL https://www.bibsonomy.org/bibtex/29a4eb97dc9592c07e19b9c4d8828a91a. [p186]

P. Delicado. Dimensionality reduction when data are density functions. *Computational Statistics and Data Analysis*, 55:401–420, 2011. URL https://doi.org/10.1016/j.csda.2010.05.008. [p186, 195]

M. Deza and E. Deza. *Encyclopedia of Distances*. Springer-Verlag, Heidelberg, 2013. URL https://doi.org/10.1007/978-3-642-00234-2_1. [p179, 183]

S. Dray, A.-B. Dufour, and D. Chessel. The **ade4** package – II: Two-table and K-table methods. *R News*, 7(2):47–52, 2007. URL https://cran.r-project.org/doc/Rnews/. [p179, 194, 201]

A. Eslami, E. M. Qannari, A. Kohler, and S. Bougeard. General overview of methods of analysis of multi-group datasets. *Revue des Nouvelles Technologies de l'Information*, 25:108–123, 2013. URL https://editions-rnti.fr/?inprocid=1001883. [p201]

A. Eslami, E. M. Qannari, S. Bougeard, and G. Sanchez. **multigroup**: *Multigroup Data Analysis*, 2020. URL https://CRAN.R-project.org/package=multigroup. R package version 0.4.5. [p201]

M. Febrero-Bande and M. O. de la Fuente. Statistical computing in functional data analysis: The r package **fda.usc**. *Journal of Statistical Software*, 51(4):1–28, 2012. URL http://www.jstatsoft.org/v51/i04/. [p179]

P. Filzmoser, K. Hron, and M. Templ. *Applied Compositional Data Analysis. With Worked Examples in R*. Springer International Publishing. Springer Nature Switzerland AG, Cham, Switzerland, 2018. ISBN 978-3-319-96420-1. Springer Series in Statistics. [p179]

G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms and Applications*. ASA-SIAM Series on Statistics and Applied Probability, SIAM,Philadelphia, ASA, Alexandria, VA, 2007. ISBN 0898716233. [p190]

C. Godin and Y. Caraglio. A multiscale model of plant topological structures. *Journal of Theoretical Biology*, 191(1):1–46, 1998. URL https://doi.org/10.1006/jtbi.1997.0561. [p199]

INSEE. Population active de 25 à 54 ans ayant un emploi et chômeurs par catégorie socioprofessionnelle et diplôme par commune et département (1968 à 2015), 2018. URL http://www.insee.fr/fr/statistiques/1893185. Accessed = 2019-06-25. [p182]

H. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14:105–122, 2000. URL https://doi.org/10.1002/1099-128X(200005/06)14:3<105::AID-CEM582>3.0.CO;2-I. [p199]

A. Kneip and K. Utikal. Inference for density families using functional principal component analysis. *Journal of the American Statistical Association*, 96:519–542, 2001. URL https://doi.org/10.1198/016214501753168235. [p186]

W. Krzanowski. *Principles of Multivariate Analysis*. Oxford University Press, New-York, 1988. ISBN 978-0198507086. [p179]

C. Lavit, Y. Escoufier, R. Sabatier, and P. Traissac. The act (statis method). *Computational Statistics & Data Analysis*, 18:97–119, 1994. URL https://doi.org/10.1016/0167-9473(94)90134-1. [p201]

S. Lê, J. Josse, and F. Husson. **FactoMineR**: An r package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008. URL https://doi.org/10.18637/jss.v025.i01. [p179, 194]

M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. **cluster**: *Cluster Analysis Basics and Extensions*, 2019. URL https://CRAN.R-project.org/package=cluster. R package version 2.1.0 — For new features, see the 'Changelog' file (in the package source). [p179, 194]

K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, London, 1979. ISBN 0-12-471252-5. [p179]

A. Petersen, P. Z. Hadjipantelis, and H. G. Mueller. **fdadensity**: *Functional Data Analysis for Density Functions by Transformation to a Hilbert Space*, 2019. URL https://CRAN.R-project.org/package=fdadensity. R package version 0.1.2. [p179]

C. Pradal and T. Cokelaer. *VPlants Documentation. Release 0.8*. INRIA, France, 2010. URL http://openalea.gforge.inria.fr/doc/vplants/newmtg/doc/_build/html/contents.html. [p199]

J. O. Ramsay, S. Graves, and G. Hooker. **fda**: *Functional Data Analysis*, 2020. URL https://CRAN.R-project.org/package=fda. R Package version 5.1.4. [p179]

J. Rudrauf and R. Boumaza. Contribution à l'étude de l'architecture médiévale: les caractéristiques des pierres à bossage des châteaux forts alsaciens. *Centre de Recherches Archéologiques Médiévales de Saverne*, 5:5–38, 2001. URL http://www.crams.fr/publications_detail.php?parPUBL=CFA05#ancDebut. [p181, 192, 197]

M. Tsagris and G. Athineou. **Compositional**: *Compositional Data Analysis*, 2020. URL https://CRAN.R-project.org/package=Compositional. R package version 3.8. [p179]

K. G. van den Boogaart, R. Tolosana-Delgado, and M. Bren. **compositions**: *Compositional Data Analysis*, 2020. URL https://CRAN.R-project.org/package=compositions. R package version 1.40-5. [p179]

W. Venables and B. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p179, 194]

M. Wand and M. Jones. *Kernel Smoothing*. Chapman and Hall, London, 1995. ISBN 9780412552700. [p185]

S. Yousfi, R. Boumaza, D. Aissani, and S. Adjabi. Optimal bandwith matrices in functional principal component analysis of density functions. *Journal of Statistical Computational and Simulation*, 85(11): 2315–2330, 2015. URL https://doi.org/10.1080/00949655.2014.928293. [p186, 194]

Rachid BOUMAZA
IRHS-UMR1345, Université d'Angers, INRAE, Institut Agro, SFR 4207 QuaSaV
49071, Beaucouzé
France
ORCiD: 0000-0002-1600-1820
rachid.boumaza@agrocampus-ouest.fr

Pierre SANTAGOSTINI
IRHS-UMR1345, Université d'Angers, INRAE, Institut Agro, SFR 4207 QuaSaV
49071, Beaucouzé
France
pierre.santagostini@agrocampus-ouest.fr

Smail YOUSFI
Département de mathématiques
Université Mouloud Mammeri
Tizi-Ouzou
Algeria
smail_yousfi@ymail.com

Sabine DEMOTES-MAINARD
IRHS-UMR1345, Université d'Angers, INRAE, Institut Agro, SFR 4207 QuaSaV
49071, Beaucouzé
France
sabine.demotes-mainard@inrae.fr

NGSSEML: Non-Gaussian State Space with Exact Marginal Likelihood

by Thiago R. Santos, Glaura C. Franco, Dani Gamerman

Abstract The number of packages/software for Gaussian State Space models has increased over recent decades. However, there are very few codes available for non-Gaussian State Space (NGSS) models due to analytical intractability that prevents exact calculations. One of the few tractable exceptions is the family of NGSS with exact marginal likelihood, named NGSSEML. In this work, we present the wide range of data formats and distributions handled by NGSSEML and a package in the R language to perform classical and Bayesian inference for them. Special functions for filtering, forecasting, and smoothing procedures and the exact calculation of the marginal likelihood function are provided. The methods implemented in the package are illustrated for count and volatility time series and some reliability/survival models, showing that the codes are easy to handle. Therefore, the NGSSEML family emerges as a simple and interesting option/alternative for modeling non-Gaussian time-varying structures commonly encountered in time series and reliability/survival studies.

Keywords: Bayesian, classical inference, reliability, smoothing, time series, software R

Introduction

Standard statistical software and packages in the State Space Model (SSM) context mainly implement Gaussian models, which are well developed in the literature. For example, the R-packages **StructTS** (Ripley, 2002), **dlm** (Petris, 2010), **dlmodeler** (Szymanski, 2014), **SSsimple** (Zes, 2019), and **MARSS** (Holmes et al., 2013) are a sample of them. The last introduces multivariate autoregressive state-space modeling or multivariate Gaussian state-space models. In spite of that, there is an increasing demand for methods or models to handle non-Gaussian and non-linear time series that can be applied to real data. In the SSM class, a general structure called Dynamic Generalized Linear Models (DGLM), proposed by West et al. (1985), attracted a great deal of interest due to its flexibility, allowing the observation distribution to belong to the exponential family. The book by Durbin and Koopman (2012) also analyzed non-Gaussian time series using importance sampling in SSM.

Many other researchers have devoted themselves, over the last decades, to auxiliary procedures to handle non-Gaussian State Space under the Bayesian approach (see, for example, Andrieu and Doucet (2002); Carvalho et al. (2010), among others). Some attempts to code NGSS include the **sspir** (Dethlefsen and Lundbye-Christensen, 2006), **pomp** (King et al., 2016), **KFAS** (Helske, 2016), **bssm** (Helske and Vihola, 2021), and **dynamichazard** (Christoffersen et al., 2021) packages in the R environment (Team, 2021) and SsfPack (Koopman et al., 1999) in the Ox software (Doornik, 1997). The SsfPack is the most consolidated package with several updated versions.

Nevertheless, even for simple structures in this class of models, the analytical tractability is easily lost, and, hence, approximations are required in order to perform inferential procedures. Thus, all packages mentioned above use intensive computational methods for making inferences for the model parameters, like sequential Monte Carlo methods or particle filters, importance sampling, particle Markov chain Monte Carlo (MCMC), etc.

In spite of this increasing number of models and packages to handle non-Gaussian SSM, a common feature shared by all of them is the need to use approximations to perform inferential procedures even in very simple cases. Gamerman et al. (2013) identified a subset of NGSS that allows for exact and analytical calculation of the marginal likelihood and the predictive and filtering distributions, called NGSSEML. This family unified several different models previously existing in the literature and included more cases. These exact computations are extremely easy to code and can be performed effortlessly and in a simple way. Applications to real data are presented, and the codes are made available by the authors in Ox language. Also, NGSSEML includes the works of Aktekin et al. (2013, 2018), de Pinho et al. (2016) for modeling count and volatility data, respectively, and was later extended by Santos et al. (2017) to include models commonly used in reliability and survival contexts.

Thus, in the face of the absence of consolidated software and packages to handle **NGSSEML**, the objective of this work is to unify the range of models encompassed and to introduce the R-package **NGSSEML**. The package illustrates how the NGSS models can be easily employed in non-Gaussian time series and reliability analysis using R under the Bayesian and classical perspectives. It also provides codes in the R environment for mainly formulating and specifying the NGSS models considered in Gamerman et al. (2013) and Santos et al. (2017). The main advantage of this package, which is designed for the specific class of NGSSEML, is that the models are implemented in a simple and flexible way. All other softwares listed above deal with state space models where the marginal likelihood can not be exactly obtained. The NGSSEML approach has the advantage of not needing to approximate the likelihood as others do, achieving that via an alternative evolution. This exact specification enables a computational gain, making the NGSSEML package an attractive option to handle non-Gaussian state space. Our approach to inference is to use (virtually) exact methods. Intensive computational methods are only required when the dimension of the unknown static parameter vector is high. In this case, the Adaptive Rejection Metropolis Sampling (ARMS) algorithm is used to approximate the marginal posterior of the static parameters (Petris, 2010). Applications to count data, volatility time series, and reliability models, such as the piecewise exponential model (PEM) and Weibull and Gamma software reliability models are presented.

The paper is organized as follows. The NGSSEML family is presented in Section 2, as well as the inferential and smoothing procedures for the latent states. Section 3 introduces the package functions. Section 4 shows illustrative examples of time series and reliability data. Finally, Section 5 presents the main conclusions and final remarks.

The model

Gamerman et al. (2013) introduced a rich class of non-Gaussian state space models with exact marginal likelihood, called here NGSSEML, based on the seminal work of Smith and Miller (1986). Inference free from approximations can be performed in the NGSSEML, and this is its main advantage compared with other NGSS procedures, such as the DGLM. Due to the model formulation, some components, such as seasonality and the effect of external covariates, are introduced as fixed effects.

A time series $\{y_t\}_{t=1}^n$ belongs to this class of models if its probability (density) function can be written as:

$$\begin{aligned} p(y_t|\mu_t, \mu_{t-1}, \ldots, \mu_1, \mathbf{Y}_{t-1}, \boldsymbol{\theta}) =& p(y_t|\mu_t, \mathbf{Y}_{t-1}, \boldsymbol{\theta}) \\ =& a(y_t, \boldsymbol{\theta})\mu_t^{b(y_t,\boldsymbol{\theta})} \exp\left(-\mu_t c(y_t, \boldsymbol{\theta})\right), \end{aligned} \tag{1}$$

for $y_t \in S(\boldsymbol{\theta}) \subset \Re$ and $p(y_t|\mu_t, \boldsymbol{\theta}) = 0$, otherwise. Functions $a(\cdot)$, $b(\cdot)$, $c(\cdot)$, and $S(\cdot)$ are such that $p(y_t|\mu_t, \boldsymbol{\theta}) \geq 0$ and, therefore, $\mu_t > 0$, for all $t > 0$. μ_t is the state at time t and $\mathbf{Y}_t = \{Y_0, y_1, \ldots, y_t\}$ represents previously available information. It is also assumed that $\boldsymbol{\theta}$ varies in the q-dimensional parameter space Θ.

The state μ_t in Equation (1) is defined as $\mu_t = \lambda_t g(\boldsymbol{x}_t, \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ are the regression coefficients (one of the components of $\boldsymbol{\theta}$), $\boldsymbol{x}_t$ is a covariate vector, and λ_t is the latent variable related to the dynamic level. The usual specification for the link function g is the logarithmic function given the positive nature of μ_t, but other link functions dictated by the application may also be used. Given λ_t, the dynamic level λ_{t+1} evolves according to the system equation

$$w\frac{\lambda_{t+1}}{\lambda_t} \mid \lambda_t, Y_t, \theta \sim \text{Beta}\left(wa_t, (1-w)a_t\right), \tag{2}$$

where a_t are values obtained in a sequential procedure and $0 < w < 1$. The parameter w is responsible for increasing the variance over time and plays a similar role to that of discount factors in the DGLM (West and Harrison, 1997). Finally, the initial prior distribution is given by $\lambda_0|Y_0 \sim \text{Gamma}(a_0, b_0)$. Thus, the full model specification is completed.

There is a wide range of distributions that belong to this class of models. It includes many commonly known discrete and continuous distributions such as Poisson, Gamma, and Normal (with static mean) but also includes many other distributions that are not so common and some reliability models. Table 1 provides the form of functions a, b, c, and S for some usual distributions in this family.

This family provides a collection of distributions for modeling a variety of real-time series and reliability data that are of practical importance, including software reliability. A detailed explanation and some illustrations can be found in de Pinho et al. (2016) and Santos et al. (2017).

Hereafter, following Theorem 1 in Gamerman et al. (2013), sequential inference for the NGSSEML will be presented as a basic result that includes the one-step-ahead predicted distributions and the filtering distribution of the latent states $\{\lambda_t\}_{t=1:n}$. If the model is defined as in (1) and (2), the following results can be obtained:

- One-step-ahead predicted (forecast) distribution of the states

  $$\lambda_t|Y_{t-1}, \boldsymbol{\theta} \sim \text{Gamma}(wa_{t-1}, wb_{t-1}), \tag{3}$$

  where $0 < w < 1$.

	Models	θ	$a(y_t,\theta)$	$b(y_t,\theta)$	$c(y_t,\boldsymbol{\theta})$
Time Series	Poisson	w,β	$(y_t!)^{-1}$	y_t	1
	Gamma	w,χ,β	$y_t^{\chi-1}/\Gamma(\chi)$	χ	y_t
	Weibull	w,ν,β	$\nu(y_t)^{\nu-1}$	1	$(y_t)^{\nu}$
	Normal	w,μ,β	$(2\pi)^{-1/2}$	$1/2$	$(y_t-\mu)^2/2$
	Laplace	w,μ,β	$\frac{1}{\sqrt{2}}$	1	$\sqrt{2}\lvert y_t-\mu\rvert$
	Power Exponential (GED*)	w,ν,μ,β	$\frac{\nu}{2^{\frac{\nu+1}{\nu}}\Gamma(1/\nu)}$	$1/\nu$	$\frac{(y_t-\mu)^{\nu}}{2}$
	Generalized Gamma	w,ν,χ,β	$\nu y_t^{\nu\chi-1}/\Gamma(\chi)$	χ	y_t^{ν}
Relia-bility	PEM⋆	w	1	$\sum_{j=1}^{N}\chi_{ij}$	$\sum_{j=1}^{N}(t_{ij}-t_{i-1})$
	PH⋆	w,β	$\exp\left(\sum_{j\in R_t}\chi_{ij}x_j'\beta\right)$	$\sum_{j=1}^{N}\chi_{ij}$	$\sum_{j=1}^{N}(t_{ij}-t_{i-1})\exp(x_{ij}'\beta)$
	Weibull SR*	w,ν,β	$\nu y_t^{\nu-1}\exp(-\nu x_t\beta)$	1	$y_t^{\nu}\exp(-\nu x_t\beta)$
	Gamma SR*	w,α,β	$\frac{(y_t)^{\alpha-1}}{\Gamma(\alpha)\exp(\alpha x_t\beta)}$	α	$y_t\exp(-x_t\beta)$

Note: ⋆PEM: Piecewise Exponential Model; PH: Proportional Hazards model; GED: Generalized Error Distribution; SR: Software Reliability.

Table 1: Special cases of the NGSSEML.

- Filtering distribution of the states

$$\lambda_t = \left(\mu_t[g(x_t'\boldsymbol{\beta})]^{-1}\right)|Y_t,\boldsymbol{\theta} \sim \text{Gamma}(a_t,b_t), \tag{4}$$

  where $a_t = wa_{t-1} + b(y_t,\boldsymbol{\theta})$ and $b_t = wb_{t-1} + c(y_t,\boldsymbol{\theta})g(x_t,\boldsymbol{\beta})$.

- One-step-ahead predictive density function of the observations

$$p(y_t|Y_{t-1},\theta) = \frac{\Gamma\left(b(y_t,\theta)+wa_{t-1}\right)a(y_t,\theta)[wb_{t-1}\left(g(x_t,\beta)\right)^{-1}]^{wa_{t-1}}}{\Gamma(a_{t-1})[c(y_t,\theta)+wb_{t-1}(g(x_t,\beta))^{-1}]^{b(y_t,\theta)+wa_{t-1}}}, \quad y_t \in S(\theta), \tag{5}$$

$\forall t \le n$ where $\Gamma(\cdot)$ is the gamma function. The distribution of $\mu_t = \lambda_t g(x_t,\beta)$ can be easily obtained from the predictive and filtering distributions.

The analytical form of the predictive density function in (5) allows the exact computation of the marginal likelihood function

$$L(\theta) = L(\theta;Y_n) = \prod_{t=1}^{n}\left(p(y_t|Y_{t-1},\theta)\right). \tag{6}$$

Thus, classical and Bayesian inference for the model parameters are easily performed.

Classical inference for the static parameter vector θ can be performed through the maximum likelihood estimator (MLE) obtained by maximizing the marginal likelihood function with respect to θ. In general, since the MLE does not possess a closed-form expression, numerical maximization methods (such as the BFGS (Shanno, 1970)) are required to maximize the likelihood function.

Bayesian inference for the static parameters can be performed combining the marginal likelihood function and a prior of the static parameters, $p(\theta)$, thus obtaining the marginal posterior

$$p(\theta|Y_n) \propto L(\theta)p(\theta). \tag{7}$$

When the posterior distribution does not have a closed-form, computational methods can be used, such as the Markov chain Monte Carlo (MCMC), Adaptive Rejection Metropolis Sampling (ARMS) (Petris, 2010), or numerical integration/quadrature (Santos et al., 2017). In the latter, an inexpensive grid of points can be utilized due to the low dimensionality of θ in many special models and examples presented in this work. Table 2 shows the prior distributions for the static parameters used in each model of Table 1.

Regarding the latent state, $\boldsymbol{\lambda} = \{\lambda_t\}_{t=1:n}$, inference can be made using its predictive and filtering distributions as well as smoothing procedures with θ set as a fixed value, e.g., its respective MLE (Brockwell and Davis, 1996), sampled from its posterior distribution via MCMC output or directly. The latter approach is explained in the sequel. In order to obtain a sample of the latent state, the smoothing procedure, described in Theorem 2 of Gamerman et al. (2013), is required. It states that the distribution

	Models	θ	**Prior** ($p(\theta)$)
Time Series	Poisson	w,β	$w \sim \text{Beta}(a_w, b_w)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Gamma	w,χ,β	$w \sim \text{Beta}(a_w, b_w)$, $\chi \sim \text{Gamma}(a_\chi, b_\chi)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Weibull	w,ν,β	$w \sim \text{Beta}(a_w, b_w)$, $\nu \sim \text{Gamma}(a_\nu, b_\nu)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Normal	w,μ,β	$w \sim \text{Beta}(a_w, b_w)$, $\mu \sim \text{Normal}(m, v)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Laplace	w,μ,β	$w \sim \text{Beta}(a_w, b_w)$, $\mu \sim \text{Normal}(m, v)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Power Exponential (GED)	w,ν,μ,β	$w \sim \text{Beta}(a_w, b_w)$, $\nu \sim \text{Gamma}(a_\nu, b_\nu)$, $\mu \sim \text{Normal}(m, v)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Generalized Gamma	w,ν,χ,β	$w \sim \text{Beta}(a_w, b_w)$, $\nu \sim \text{Gamma}(a_\nu, b_\nu)$, $\chi \sim \text{Gamma}(a_\chi, b_\chi)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
Relia-bility	PEM*	w	$w \sim \text{Beta}(a_w, b_w)$
	PH*	w,β w,β	$w \sim \text{Beta}(a_w, b_w)$, $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Weibull SR	w,ν,β	$w \sim \text{Beta}(a_w, b_w)$, $\nu \sim \text{Gamma}(a_\nu, b_\nu)$ $\beta \sim \text{Normal}_q(m_{beta}, V_{beta})$
	Gamma SR	w,α,β	$w \sim \text{Beta}(a_w, b_w)$, $\alpha \sim \text{Gamma}(a_\alpha, b_\alpha)$ $\beta \sim \text{Normal}_q((m_{beta}, V_{beta})$

Note: *PEM: Piecewise Exponential Model; PH: Proportional Hazards model; GED: Generalized Error Distribution; SR: Software Reliability.

Table 2: Prior for the static parameters of the NGSSEML.

of $(\boldsymbol{\lambda}|\boldsymbol{\theta}, Y_n)$ is given by

$$p(\boldsymbol{\lambda}|\boldsymbol{\theta}, Y_n) = p(\lambda_n|\boldsymbol{\theta}, Y_n) \prod_{t=1}^{n-1} p\left(\lambda_t|\lambda_{t+1}, \boldsymbol{\theta}, Y_t\right),$$

where

$$\lambda_t - w\lambda_{t+1}|\lambda_{t+1}, \boldsymbol{\theta}, Y_t \sim \text{Gamma}\left((1-w)a_t, b_t\right), \forall t \geq 0. \tag{8}$$

In an analogous way, the marginal distribution of $(\boldsymbol{\lambda}|Y_n)$ is expressed as

$$p(\boldsymbol{\lambda}|Y_n) = \int p(\boldsymbol{\lambda}|\boldsymbol{\theta}, Y_n) p(\boldsymbol{\theta}|Y_n) d\boldsymbol{\theta}. \tag{9}$$

Once a sample $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ is available, samples from the states are obtained retrospectively from the distribution of the states using the result in (8). More details can be found in Gamerman et al. (2013). A sample $\boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(M)}$ of the marginal distribution in (9) is obtained in the following manner: i) sample $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}$ from its marginal posterior $p(\boldsymbol{\theta}|Y_n)$. ii) sample $\boldsymbol{\lambda}^{(j)}$ from $p(\boldsymbol{\lambda}|\boldsymbol{\theta}^{(j)}, Y_n)$, $j = 1, \ldots, M$. If the dimensionality of the static parameters θ is low, the numerical calculation of $p(\theta|Y_n)$ using quadrature is indistinguishable from the exact calculation. If θ is discrete, it is an exact result. Thus, the numerical calculation of the marginal posterior distribution $p(\boldsymbol{\lambda}|Y_n)$ is a suitable approximation, whose quality depends on the approximation of $p(\theta|Y_n)$ and, hence, the dimensionality of θ and its nature.

Gamerman et al. (2013) used the Pearson and deviance residuals for model diagnostics and AIC, BIC, and DIC for model comparison in this family. It should be noted that, if MCMC methods are used, convergence checks using graphs like trace plot, autocorrelogram, etc., and tests like the Geweke and Gelman-Rubin tests are necessary.

Specification of the state space objects

The package NGSSEML can be downloaded and installed from GitHub or CRAN website and is then activated in R by `library("NGSSEML")`. Assuming that the data are available in the current environment, the NGSSEML can be set up using the formulas and family arguments presented in the next subsections.

Estimation of the static parameters

This subsection presents the estimation of the static parameters $\boldsymbol{\theta}$ under classical and Bayesian approaches. The classical procedure uses the BFGS algorithm, implemented in the "optim" function of the R package, to obtain the MLE. The Bayesian procedure uses numerical integration/quadrature with an inexpensive grid of points. The functions 'ngssm.mle' and 'ngssm.bayes' perform the classical and Bayesian inferences, respectively, shown in the previous section and utilize the function 'LikeF' (the marginal likelihood function) in Eq. (6), internally.

Basically, the functions of NGSSEML work with a very simple specification of a few arguments. We need to specify the `formula`, `data` (data frame) arguments, and `model`, as well as the function 'lm' for linear regression. For piecewise exponential models in reliability analysis, we cannot forget to include the `breaks` and `events` arguments.

The function 'ngssm.mle' performs the marginal likelihood estimation of the static parameters of the model using Eq. (6), and can be executed by the following command:

```
> ngssm.mle(formula, data, na.action="na.omit", pz = NULL, nBreaks = NULL,
+ model = "Poisson", StaPar = NULL, amp = FALSE, a0 = 0.01, b0 = 0.01,
+ ci = 0.95, LabelParTheta = NULL, verbose = TRUE, method = "BFGS",
+ hessian = TRUE, control = list(maxit = 30000, temp = 2000,
+ trace = FALSE, REPORT = 500))
```

The output of the model fit presents the maximum likelihood estimators, standard errors, Z statistics, and asymptotic confidence intervals of the model parameters. Furthermore, it allows the user to change the control settings, choose the optimizing algorithm, and set arguments as the hyperparameters a_0 and b_0, the confidence level, etc.

The function 'ngssm.bayes' performs the Bayesian estimation of the static parameters of the model using the marginal posterior in Eq. (7), and can be run by the following command:

```
> ngssm.bayes(formula, data, na.action = "na.omit", pz = NULL, nBreaks = NULL,
+ model = "Poisson", StaPar = NULL, amp = FALSE, a0 = 0.01, b0 = 0.01, prw = c(1, 1),
+ prnu = NULL, prchi = NULL, prmu = NULL, prbetamu = NULL, prbetasigma = NULL,
+ lower = NULL, upper = NULL, ci = 0.95, pointss = 10, nsamplex = 1000, mcmc = NULL,
+ postplot = FALSE, contourplot = FALSE, LabelParTheta = NULL, verbose = TRUE)
```

The model fit's output provides the Bayesian estimators, posterior mean and median, standard error, and percentile credibility intervals for NGSSEML parameters, and the posterior samples of the static parameters of the NGSSEML using multinomial sampling. The main functions of the model fit provide an object class "ngssm", associated with print, summary, and extract further information. Besides, the user can choose the number of grid points (the argument `pointss`) of the quadrature and samples (the argument `nsamplex`) and set the hyperparameters of prior distributions (the arguments `a0,b0,prw,prnu,prchi,prmu,prbetamu,prbetasigma`), the credibility level, etc. If `mcmc=TRUE` or the number of points in the grid is very high, the ARMS algorithm is executed instead of the quadrature method. Samples of the posterior distribution of the latent states using the smoothing procedure are calculated as described in Eq. (9).

Estimation of latent states

The filtering, smoothing, and plot functions for the NGSSEML are shown in this subsection. For these functions, the static parameters are estimated with the whole data.

The function 'FilteringF' computes the shape and scale parameters of the one-step-ahead forecast and filtering distributions of the latent states using the filters in Eq. 3 and 4, respectively, for several models. It can be called by the command below.

```
> FilteringF(formula, data, na.action = "na.omit", pz = NULL, nBreaks = NULL,
+ model = "Poisson", StaPar = NULL, a0 = 0.01, b0 = 0.01, amp = FALSE,
+ distl = "PRED", splot = FALSE)
```

The output presents the shape and scale parameters of the one-step-ahead forecast and filtering distributions of the latent states. Furthermore, it allows the user to modify the arguments of the function as the hyperparameters a_0 and b_0, etc.

The function 'SmoothingF' provides an exact sample of the smoothing distribution of the latent states in Eq. (9), and its command is

```
> SmoothingF(formula, data, na.action = "na.omit", pz = NULL, nBreaks = NULL,
+ model = "Poisson", StaPar = NULL, Type = "Cond", a0 = 0.01, b0 = 0.01,
+ amp = FALSE, samples = 1, ci = 0.95, splot = FALSE)
```

The function provides an object with an exact sample of the joint distribution of the states. The user can choose the arguments as the hyperparameters a_0 and b_0, type of the distribution of the latent states - margin/conditional on the static parameters, the confidence/credibility level, etc. If the number of samples is greater than 1, some summaries of the state samples are returned. If the argument `splot=TRUE`, a graph with the states' smoothed estimates is shown.

The function 'PlotF' builds graphs with smoothed/filtered estimates of the latent states and can be run by the following command:

```
> PlotF(formula, data, na.action = "na.omit", pz = NULL, nBreaks = NULL,
+ plotYt = TRUE, axisxdate = NULL, transf = 1, model = "Poisson", posts,
+ Proc = "Smooth", Type = "Marg", distl = "PRED", a0 = 0.01, b0 = 0.01,
+ ci = 0.95, startdate = NULL, enddate = NULL, Freq = NULL, ...)
```

The function returns a graph with smoothed or filtered estimates of the latent states. The user can change the arguments as the hyperparameters a_0 and b_0, the type of the distribution of the latent states - margin/conditional on the static parameters, the procedure between smoothing and filtering, the confidence/credibility level, etc. Other options related to graph edition, such as color, type line, labels, can be inserted into the function 'PlotF' using the argument ellipsis (...). To save space in this manuscript, they are not shown. We will try to provide more details in the examples.

Examples

In this section, we present four examples to illustrate the use of the proposed package for count, volatility, lifetime, and software reliability data. The count time series is the poliomyelitis data in the USA. The volatility series refers to the return data set from the Petrobras stock market. The lifetime data set (GTE) are the daily failure times of 125 telecommunication systems. The "SYS1" data set are times between 136 successive computer software failures. All examples are widely used in the literature of their respective areas. We present parameter estimation, the goodness of fit for the adjusted models and predictions. For the poliomyelitis data, we also provide comparisons with other softwares available in the literature.

As previously explained, this software is designed for the specific class of non-Gaussian state space models with exact marginal likelihood. The NGSSEML may seem analytically more complex in comparison with other procedures, which present in general simpler equations. Nevertheless, although simplest in form, these procedures frequently need more calculations and heavier computational effort to produce similar results to our method. Based on an alternative evolution, our approach requires less approximation, and thus, inference for static parameters is almost indistinguishable from the (unobtainable) exact form. Readers interested in the advantages and comparisons of the NGSSEML against alternatives should refer to Gamerman et al. (2013) and Santos et al. (2017).

Count data

A Poisson NGSSEML model is used to fit the monthly data of the number of poliomyelitis cases in the USA, shown in Figure 1, from 1970/01 to 1983/12 (168 observations). The polio data is a well-known example in the count time series literature and was first analyzed by Zeger (1988). The objective is to explain the number of poliomyelitis cases through covariates and make predictions. The covariates are the deterministic trend centered at 73, annual cosine and sine, and semiannual cosine and sine. The intercept is inserted in the model as a dynamic level. The confidence and credibility levels for the intervals are fixed at 0.95. The state initial condition is $\lambda_0|Y_0 \sim \text{Gamma}(a_0, b_0)$, where $a_0 = 0.3$ and $b_0 = 0.1$ that can be specified in the functions 'ngssm.mle' and 'ngssm.bayes' by the arguments $a0$ and $b0$. The static parameter vector in this example is $\boldsymbol{\theta} = (w, \beta_1, \beta_2, \beta_2, \beta_3, \beta_4, \beta_5)$, so it is necessary to specify a joint probability density for it, that is, a joint prior $p(\boldsymbol{\theta})$. Assuming that the static parameters are independent and based on the priors available for the package in Table 2, the marginal priors for its components are $w \sim \text{Beta}(1,1)$ and $\beta_j \sim N(0,10)$, for $j = 1,\ldots,4$. Parameter w controls the information loss over time, while parameters β are regression coefficients associated with the covariates for capturing trend and seasonal patterns of the count time series. The prior hyperparameters are chosen to set vague priors for the model parameters.

The MLE for the static parameters of the NGSSM in the poliomyelitis data can be obtained using the BFGS algorithm (default: `method = "BFGS"`) implemented in the optim function, which is the default of 'ngssm.mle' function. It is necessary to specify `data1 = data.frame(Ytm,Xtm)` in the argument data, the formula `Ytm ~ CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual`, and the model `model = "Poisson"`. The arguments `StaPar = c(0.79,-0.11,-0.49,0.18,-0.38)`, `a0 = 0.01`, `b0 = 0.01`, and `ci = 0.95` are optional. The name of the variables in the argument must coincide with the name in the data. The model fit is stored in the `fit` object. The code for classical inference is given below. We should note that a graph with the filtered estimates of the latent states is returned when the 'FilteringF' function is called.

```
> library(NGSSEML)
> data(Polio_data)
> Xtm = Polio_data[, 3:7]
> Xtm[,1] = (1:168-73)/168
> Ytm = Polio_data$y
> Ztm=NULL
## CosAnnual, SinAnnual, CosSemiAnnual, SinSemiAnnual
## LabelParTheta = c("w", "Beta1", "Beta2", "Beta3", "Beta4")
StaPar = c(0.79, -0.11, -0.49, 0.18, -0.38)
> a0 = 0.2
> b0 = 0.1
> ci = 0.95
> data1=data.frame(Ytm,Xtm)
## Fit
> fit = ngssm.mle(Ytm ~ CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual,
+ data = data1, model = model, pz = NULL, StaPar = StaPar,
+ a0 = a0, b0 = b0, ci = ci)
> esttheta = as.numeric(fit[,1])
> filt = FilteringF(Ytm ~ Trend + CosAnnual + SinAnnual + CosSemiAnnual
+ SinSemiAnnual, data = data1, StaPar = esttheta,
+ model = "Poisson", a0 = a0, b0 = b0,
+ distl = "FILTER", splot = TRUE)
> filtest = (filt[1,]/filt[2,])
```

The results are presented in Table 3, showing the MLE's and their respective 95% confidence intervals for w and $\beta's$. Bayesian estimation can be performed using quadrature rules with a grid of 6 points (`pointss = 6`), providing a sample of 2,000 draws (`nsamplex = 2000`) of the marginal posterior distribution of the static parameters. This can be performed without the use of intensive computational methods to approximate the posterior distribution as, is usually the case in other non-Gaussian state-space models. We should set the arguments in the function as the data frame `data1`, the formula `Ytm ~ CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual`, and the model `model = "Poisson"`. The marginal priors are: $w \sim U(0,1) \equiv \text{Beta}(1,1)$, and $\beta_j \sim N(0,10)$, for $j = 1,\ldots,4$, specified in the function 'ngssm.bayes' by the following arguments `prw = c(1,1)` (two shape parameters of a Beta distribution), `prbetamu = rep(0,4)`, and `prbetasigma = diag(10,4,4)` (two parameters of a multivariate Normal distribution). If `postplot = TRUE` and `contourplot = TRUE`, the graph of the marginal posterior and the contour plot are provided. If `verbose = TRUE`, the estimation results are shown; otherwise, they are omitted. The code is presented below.

```
> library(NGSSEML)
> data(Polio_data)
> Xtm=Polio_data[,3:7]
> Xtm[,1] = (1:168-73)/168
> Ytm = Polio_data$y
> Ztm = NULL
## CosAnnual, SinAnnual, CosSemiAnnual, SinSemiAnnual
## LabelParTheta=c("w", "Beta1", "Beta2", "Beta3", "Beta4")
> StaPar = c(0.79, -0.11, -0.49, 0.18, -0.38)
> a0 = 0.2
> b0 = 0.1
## points
```

```
> pointss = 6
## posterior sample
> nsamplex = 2000
## Cred. level
> ci = 0.95
> data1 = data.frame(Ytm, Xtm)
## Bayesian fit
> fitbayes = ngssm.bayes(Ytm ~ CosAnnual + SinAnnual +
 CosSemiAnnual + SinSemiAnnual,
+ data = data1, model = "Poisson", pz = NULL, StaPar = StaPar, a0 = a0, b0 = b0,
+ prw = c(1, 1), prbetamu = rep(0, 4), prbetasigma = diag(10, 4, 4), ci = ci,
+ pointss = pointss, nsamplex = nsamplex, postplot = TRUE, contourplot = TRUE,
+ verbose = TRUE)
```

The results of the Bayesian inference are also presented in Table 3, showing the posterior mean and the credibility intervals. The coefficient regression related to the determinist trend is not significant at the 5% significance level, so we decide to remove it from the model. Analyzing Table 3, we can see that both for β_2 and β_4 associated with the trigonometric annual and semiannual sine covariates, respectively, are negative and seem to be significant under both approaches at the same significance level. We keep the cosine covariates to preserve the trigonometric representation of seasonality. The magnitude and significance of the regression coefficients are similar to those obtained by other works in the count time series literature (Davis and Wu, 2009). Furthermore, the classical estimates are close to the Bayesian estimates. If the ARMS algorithm is used, we need to make a convergence check using graphs like trace plot, autocorrelogram, etc., and tests like the Geweke and Gelman-Rubin tests in the R-package **coda**.

The R codes for the smoothing function below are used to build the graphs presented in Figure 1, for the Polio data with the smoothed estimates for the observations and mean under the Bayesian approach. The first three arguments of the function 'PlotF' refer to the Bayesian model obtained with 'ngssm.bayes'. The commands `posts = posts`, `axisxdate = x`, `Proc = "Smooth"`, `Type = "Marg"`, `startdate = "1970/01/01"`, `enddate = "1983/12/31"`, and `Freq = "months"` indicate, respectively, the sample of the marginal posterior of the static parameters, the vector of monthly dates, the smoothed distribution for the states, the chosen distribution for the latent states, the start and end dates, and the frequency of the observations. The last arguments refer to the type of lines, colors, and legends in the graph. We can see that the smoothed mean trajectory in Figure 1 follows well the behavior of the time series.

```
> posts = fitbayes$samplepost
> x = seq(as.Date("1970/01/01"), as.Date("1983/12/31"), "months")
> PlotF(Ytm ~ CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual,
+ data = data1, model = "Poisson", axisxdate = x, Proc = "Smooth", Type = "Marg",
+ posts = posts, startdate = "1970/01/01", enddate = "1983/12/31",
+ Freq = "months", type = 'l', col = c("black", "blue","lightgrey"), xlab = "Months",
+ ylab = "The number of poliomyelitis cases", xlim = NULL, ylim = c(0, 15),
+ Lty = c(1, 2, 1), lwd = c(2, 2, 2), cex = 0.68)
```

θ	MLE	95% Conf. Int.	Posterior Mean	95% Cred. Int.
w	0.793	[0.711; 0.874]	0.783	[0.713; 0.843]
β_1	-0.116	[-0.324; 0.092]	-0.117	[-0.314; 0.015]
β_2	-0.488	[-0.717; -0.259]	-0.491	[-0.707; -0.344]
β_3	0.175	[-0.024; 0.374]	0.176	[-0.014; 0.301]
β_4	-0.385	[-0.577; -0.193]	-0.387	[-0.568; -0.265]

Table 3: Point and interval (level 95%) estimation for the Poisson model fitted to the polio count data.

Figure 2 shows the standardized Pearson residuals and its autocorrelogram for model diagnostic. The residuals are not correlated and distributed between -2 to 2, except the points indicated in the graph (a) that coincide with the largest values of the series in Figure 1. Further investigation of these points could be performed, and the inclusion of interventions may improve the model fit. The R code for the diagnostic analysis in this example is

```
> esttheta = as.numeric(fit[,1]); n = length(Ytm)
> filt = FilteringF(Ytm ~ CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual,
```

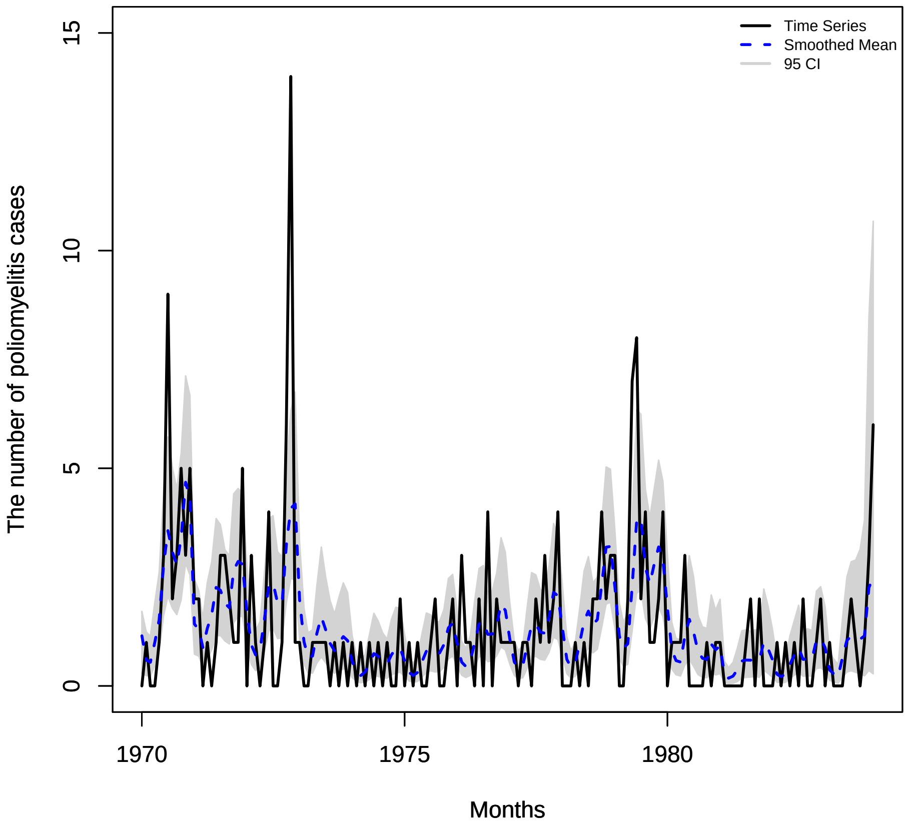


Figure 1: The polio data: The full line represents the polio monthly time series in the USA from 1970/01 to 1983/12 (168 observations). The dashed line and the grey area indicate the smoothed means and their respective 95% percentile credibility intervals using the smoothing procedure and the quadrature method.

```
+ data = data1, StaPar = esttheta, model = "Poisson", distl = "FILTER",
+ splot = TRUE)
> filtest = (filt[1,]/filt[2,])
> pred = FilteringF(Ytm ~CosAnnual + SinAnnual + CosSemiAnnual + SinSemiAnnual,
+ data = data1, StaPar = esttheta, model = "Poisson", a0 = a0, b0 = b0,
+ distl = "PRED")
> predonestepahead = (pred[1,]/pred[2,])
> residuals = (Ytm-predonestepahead)/sqrt(predonestepahead)
> summary(residuals)
> residuals = scale(residuals[1:n])
> par(mfrow = c(1, 2))
> x = seq(as.Date("1970/01/01"), as.Date("1983/12/31"), "months")
> plot(x[1:n], residuals, xlab = "Months", ylab = "Standardized Pearson residuals")
> points(x[7], residuals[7], col = "red", lwd = c(2))
> points(x[35], residuals[35], col = "red", lwd = c(2))
> points(x[74], residuals[74] ,col = "red", lwd = c(2))
> points(x[113], residuals[113], col = "red", lwd = c(2))
> title("(a)")
> acf(residuals, ci = 0.99, main = "(b)")
```

For comparison purposes, Table 4 displays some statistics for evaluating the in-sample performance of the fitted model using the **KFAS**, **sspir** and our software. The statistics are the square root of the mean square error SMSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, the mean absolute error MAE $= \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, and the mean absolute percentage error, MAPE $= \frac{1}{n}\sum_{i=1}^{n}|(y_i - \hat{y}_i)/y_i|$, where $\hat{y}_i$, $i = 1, \ldots, n$, are the fitted values. The Poisson model with the logarithmic link function, level component, and covariates is specified

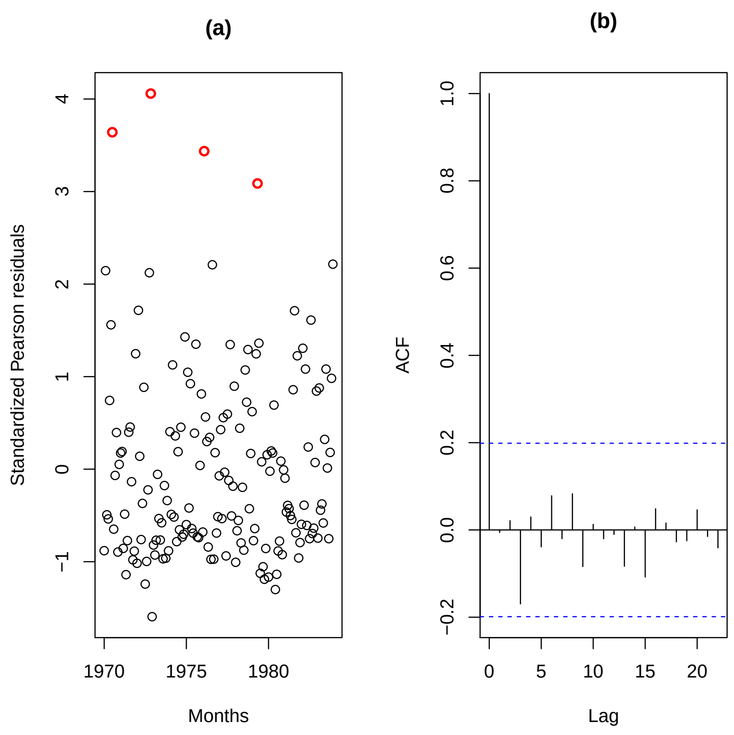


Figure 2: Polio data. (a) The full line represents the standardized Pearson residuals. The marked points are the largest values of the residuals; (b) The autocorrelogram of the standardized Pearson residuals. The marked points indicate the highest residual values.

for the three packages, as described in this subsection. The static parameters were fixed at the MLE and posterior mean for our package and the MLE for the other packages to get the smoothed/filtered mean. The results are similar for the KFAS and sspir packages, which use the maximum likelihood estimation via importance sampling methods. Our package provided the predicted values (the filtered estimates) using the classical and Bayesian approaches described in this paper and presented the best results for the SMSE, MAE, and MAPE. Convergence of the maximization algorithm is fast and stable because the likelihood is a well-behaved and exact function. The joint likelihood function (6) of our Poisson model is a product of the negative binomial distributions given by Eq. (5).

	SMSE	**MAE**	**MAPE**
KFAS*	1.776	1.179	0.638
NGSSEML*	1.280	0.890	0.468
NGSSEML**	1.264	0.880	0.462
sspir*	1.788	1.202	0.648

Note: *MLE; **The posterior mean.

Table 4: In sample SMSE, MAE, and MAPE for the Poisson model using the KFAS, NGSSEML, and sspir packages.

Volatility data

The second example refers to the daily return data from Petrobras stock market from 2000/01/06 to 2008/29/01 (1999 observations), which can be obtained on the website finance.yahoo.com/. The return at time t is defined as $y_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$, where P_t is the daily closing spot price. Data collection irregularity due to holidays and weekends will be ignored. Figure 3 presents the time series plot of y_t. A distinctive feature of financial series is that they usually present non-constant variance or volatility. Thus, this series can be modeled using distributions with a heavy tail, such as GED, which does not belong to the exponential family. The aim is to estimate the daily volatility of Petrobras return time series.

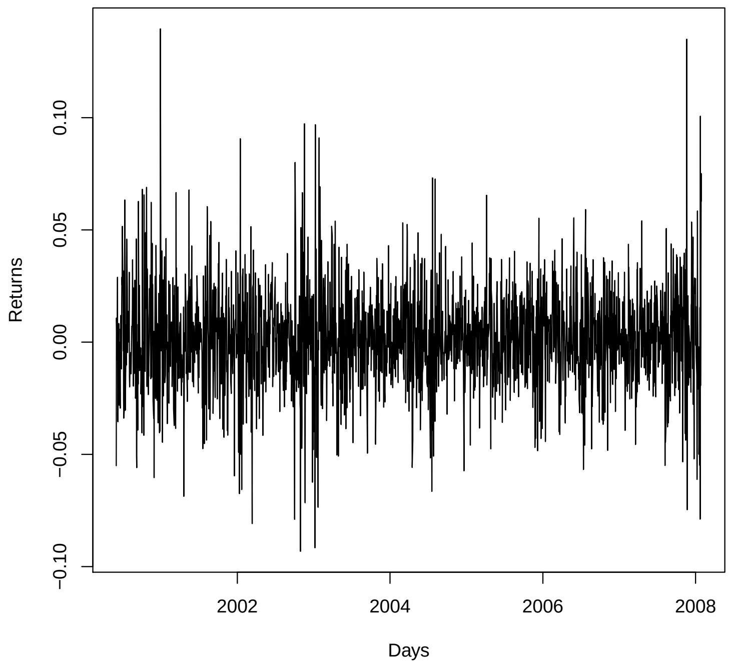


Figure 3: Petrobras return data: This time series is the Petrobras asset's daily return data in the Brazilian stock market from 2000/01/06 to 2008/29/01 (1999 observations).

The maximum likelihood estimation for the Petrobras return data is performed using the R code below with the following arguments in the function 'ngssm.mle': `Ytm~1`, `data = data.frame(Ytm)`, and `model = "GED"`. The state initial condition is $\lambda_0|Y_0 \sim \text{Gamma}(a_0, b_0)$, where $a_0 = b_0 = 0.01$ (the default values) that can be specified in the functions 'ngssm.mle' and 'ngssm.bayes' by the arguments $a0$ and $b0$.

```
> library(NGSSEML)
> data(Return_data)
> plot(Return_data[,2], Return_data[,1], type = 'l', xlab = "Days",
+ ylab = "Returns")
> Ytm = Return_data$Rt
> Xt = NULL
> Zt = NULL
> model = "GED"
## LabelParTheta = c("w", "nu")
> StaPar = c(0.9, 1)
> a0 = 0.01
> b0 = 0.01
> ci = 0.95
> fit = ngssm.mle(Ytm ~ 1, data=data.frame(Ytm), model = model, a0 = a0, b0 = b0,
+ ci = ci, verbose = FALSE)
```

Bayesian estimation for the Petrobras return data is obtained via quadrature with 15 points (argument `pointss=15`) and a sample of 1000 draws (`nsamplex=1000`) of the marginal posterior distribution. The state initial condition is $\lambda_0|Y_0 \sim \text{Gamma}(0.01, 0.01)$. Analogously to the previous example, the first arguments, which refer to the marginal prior distribution (see Table 2) for the static parameter vector $\boldsymbol{\theta} = (w, \nu)$ are $w \sim U(0, 1) \equiv \text{Beta}(1, 1)$ and $\nu \sim \text{Gamma}(0.01, 0.01)$, which can be specified by the arguments `prw = c(1,1)` (two shape parameters of a Beta distribution) and `prmu=rep(0.01,0.01)` (two parameters of a Gamma distribution with the mean equals to one). The difference here is that `model = "GED"`. Again, the prior hyperparameters are chosen to set vague priors for the model parameters. A summary of the estimates was contained in the objects `fitbayes` (Bayesian fit). The code is presented below.

```
> library(NGSSEML)
```

θ	MLE	95%Conf. Int.	Posterior Mean	95% Cred. Int.
w	0.949	[0.931; 0.966]	0.947	[0.928; 0.962]
ν	1.601	[1.458; 1.745]	1.606	[1.467; 1.743]

Table 5: Point and interval estimation for the GED model fitted to the Petrobras series.

```
> data(Return_data)
> Ytm = Return_data$Rt
> Date = Return_data$Date
> Xtm = NULL
> Ztm = NULL
## LabelParTheta = c("W", "nu")
> StaPar =c (0.9,1)
> a0 = 0.01
> b0 = 0.01
## points
> pointss = 15
## posterior sample size
> nsamplex = 1000
## Cred. level
> ci = 0.95
## Bayesian fit
> fitbayes = ngssm.bayes(Ytm ~ 1, data = data.frame(Ytm), model = "GED", pz = NULL,
+ StaPar = StaPar, a0 = a0, b0 = b0, prw = c(1, 1), prnu = c(0.01, 0.01), ci = ci,
+ pointss = pointss, nsamplex = nsamplex, postplot = TRUE, contourplot = TRUE,
+ verbose  =TRUE)
```

Table 5 presents the point and the interval estimates for the GED model fitted to the Petrobras return data under the classical and Bayesian approaches obtained by the above codes. Figure 4 shows the marginal posterior distributions and contour plot of the joint posterior distribution of the parameters w and ν, respectively, with `postplot = TRUE` and `contourplot = TRUE` in the code of the Bayesian approach. Parameter w controls the information loss over time, while r is the shape parameter of the GED model and controls its tails. If $\nu < 2$, the GED is a heavy-tailed distribution. It is the case of the Petrobras return data, as expected, since the estimate for ν is around 1.6 and the upper limit of the 95% intervals for ν is smaller than 2.

To illustrate the usage of the 'PlotF' function in this example, a detailed description of its arguments is given. The first three entries in the 'PlotF' function refer to the data and model used. As there are no explanatory variables to be inserted in the volatility, `pz = NULL`. A date vector for the x-axis was specified in the `axisxdate` argument. The time series was not inserted in the plot, thus `plotYt = FALSE`. A transformation of -0.5 was applied to the estimates of the latent states to get the volatility. The chosen distribution for the latent states was conditional on the static parameters and marginal (`Type = "Marg"`). A sample of the static parameters was set as `posts = fitbayes$samplepost`, and the latent states distribution was the smoothed (`Proc = "Smooth"`). The `started` date was set as `'2000-06-01'` and the `enddate` as `'2008-01-29'`. The frequency of data is daily (`Freq = "days"`). The remaining arguments refer to type of lines, colors, and legends in the graph. We can see that the peaks in Figure 5 correspond to periods of crisis known in the literature and are pointed out by the model.

```
## Smoothing
> set.seed(1000)
> posts = fitbayes$samplepost
> PlotF(Ytm ~ 1, data = data.frame(Ytm), model = "GED", pz = NULL,
+ axisxdate = Return_data$Date, plotYt = FALSE, transf = -0.5, Proc = "Smooth",
+ Type = "Marg", posts = posts, startdate = '2000-06-01', enddate = '2008-01-29',
+ Freq = "days", typeline = 'l', col = c("black", "blue", "lightgrey"),
+ xlab = "Days", ylab = expression(paste(hat(sigma)[t])), xlim = NULL,
+ ylim = c(0.02,0.10), lty = c(1, 2, 1), lwd = c(2, 2, 2), cex = 0.68)
```

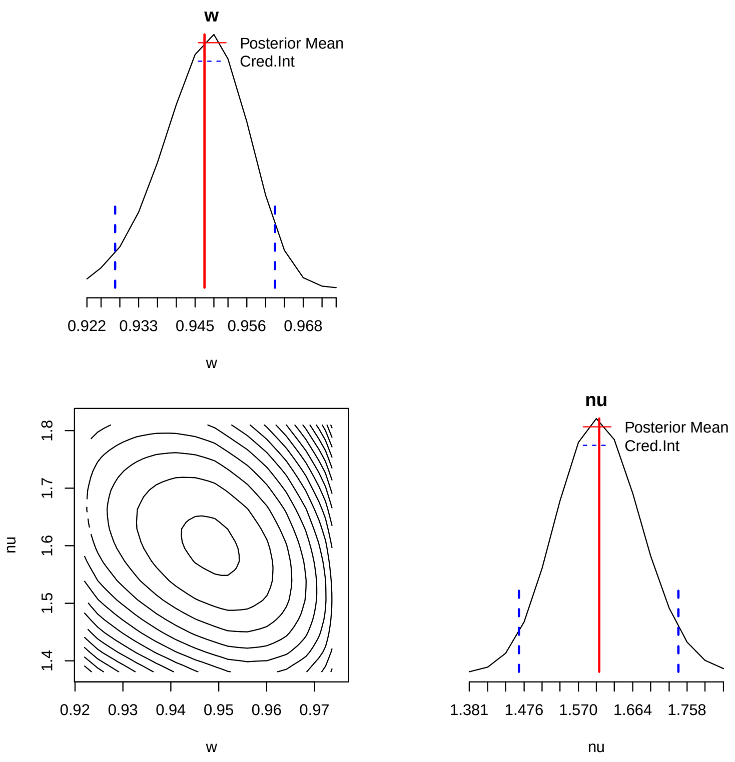


Figure 4: Petrobras return data: The marginal posterior distribution of the parameters w and ν and their respective contour plot, obtained by the quadratures.

Lifetime data

The third example refers to the daily failure times of 125 telecommunication systems installed by the GTE corporation in a pre-specified time period (Kim and Proschan, 1991). Seventeen failures were observed out of the 125 observations. The purpose is to estimate the failure rate for the GTE data.

We specify the formula `Ytm ~ Event` in the first argument of 'ngssm.mle' function. The data must contain columns with the times and events. We should also define the model (`model = "PEM"`) and obtain the breaks (one per interval) using the function 'GridP' with the times, events, and `nT = NULL` (the number of one interval per failure). The breaks are automatically inserted into the functions 'ngssm.mle' and 'ngssm.bayes' with `nBreaks = NULL`. The maximum likelihood estimation for the GTE data is performed using the R code below.

```
> library(NGSSEML)
> data(gte_data)
## Times
> Ytm = gte_data$V1
> Xtm = NULL
> Ztm = NULL
> amp = FALSE
## Event: failure, 1
> Event = gte_data$V2
> Break=NGSSEML:::GridP(Ytm, Event, nT = NULL)
## LabelParTheta = c("w")
> StaPar = c(0.73)
> a0 = 0.01
> b0 = 0.01
> ci = 0.95
> fit = ngssm.mle(Ytm ~ Event, data = data.frame(Ytm,Event), model = "PEM",
+  nBreaks = NULL, amp = amp, a0 = a0, b0 = b0, ci = ci, verbose = FALSE)
```

Bayesian estimation for the GTE data is built using quadrature with a grid of 50 points (`pointss = 50`) and a sample of 1000 draws (`nsamplex = 1000`). The state initial condition is $\lambda_0|Y_0 \sim \text{Gamma}(a_0, b_0)$, where $a_0 = b_0 = 0.01$ are specified in the functions by the arguments $a0$ and

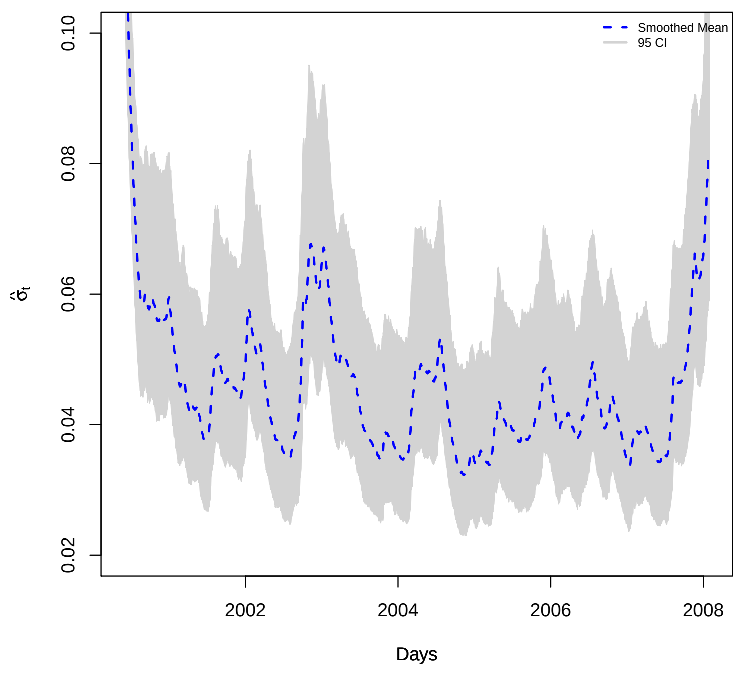


Figure 5: Petrobras return data: The dashed and dotted lines represent the smoothed estimate and the one-step-ahead prediction of the stochastic volatility ($\hat{\sigma}_t = \hat{\lambda}_t^{1/\hat{\nu}}$), obtained by the fit of the GED model under the Bayesian approach. The grey area indicates the 95% percetile credibility interval.

$b0$, and the prior is $w \sim U(0,1) \equiv \text{Beta}(1,1)$. Both priors are specified in the function 'ngssm.bayes' in a similar manner of the previous examples to get vague priors. The formula `Ytm ~ Event` contains the times and events, and the breaks are automatically inserted with `nBreaks = NULL`. The R code is below.

```
> library(NGSSEML)
> data(gte_data)
## Times
> Ytm = gte_data$V1
## Event: failure, 1
> Event = gte_data$V2
> Break = NGSSEML:::GridP(Ytm, Event, nT = NULL)
> Xtm = NULL
> Ztm = NULL
> amp = FALSE
## LabelParTheta = c("w")
> StaPar = c(0.5)
> lower = c(0.01)
> upper = c(0.99)
> a0 = 0.01
> b0 = 0.01
## points
> pointss = 50
## Number of samples from the posterior
> nsamplex = 1000
## Bayesian fit
> fitbayes = ngssm.bayes(Ytm ~ Event, data = data.frame(Ytm,Event), model = "PEM",
+ StaPar = StaPar, amp = amp, a0 = a0, b0 = b0, prw = c(1,1), pointss = pointss,
+ nsamplex = nsamplex, postplot = TRUE, contourplot = FALSE, verbose = TRUE)
```

Table 6 presents the point and the interval estimates for the PEM fitted to the GTE data under the classical and Bayesian approaches. Parameter w controls the information loss over time in the same manner as the previous application. If `pz = TRUE`, w becomes w_i for each interval, weighted by the interval width. These estimates are stored in the objects "fit" (classical fit) and "fitbayes" (Bayesian fit) of the previous codes.

The code below is used to build Figure 6, which shows the smoothed failure rate estimates under the Bayesian approach, with a 95% credibility interval (the grey area). As already mentioned, the first three entries in the 'PlotF' function refer to the data and model used. The date for the x-axis was specified in `axisxdate = Break[1:17]`. Once more, the chosen distribution for the latent states was conditional on the static parameters and marginal (`Type = "Marg"`), and the latent states distribution was the smoothed (`Proc = "Smooth"`). A sample of the static parameters was built as posts=fitbayes$samplepost. The remaining arguments refer to type of lines, colors, and legends in the graph. We can note that the failure rate is decreasing, as expected.

```
> posts = fitbayes$samplepost
> set.seed(1000)
> PlotF(Ytm ~ Event, data = data.frame(Ytm, Event), axisxdate = Break[1:17],
+ model = "PEM", Proc = "Smooth", Type = "Marg",  posts = posts, typeline = 's',
+ col = c("black", "blue", "lightgrey"),  xlab = "Time to Failure (Days)",
+ ylab = "Failure rate", xlim = c(0, 139), ylim = c(0, 0.008), lty = c(1, 2, 1),
+ lwd = c(2, 2, 2), cex = 0.68)
```

θ	MLE	95%Conf. Int.	Posterior Mean	95% Cred. Int.
w	0.773	[0.518; 1.000]	0.752	[0.503; 0.956]

Table 6: Point and interval (level 95%) estimation for the PEM fitted to the GTE data.

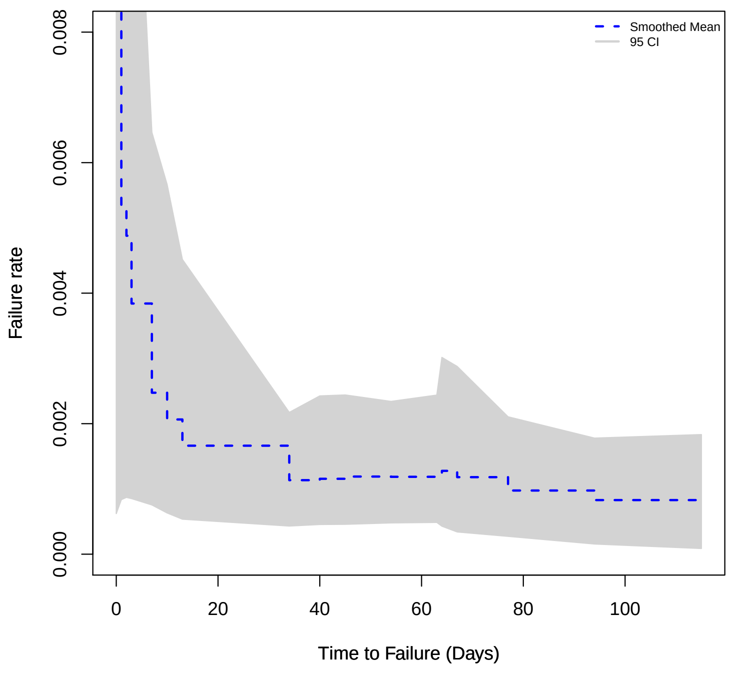


Figure 6: The full line indicates the smoothed estimate of the failure rate for the GTE data, and the grey area is its respective 95% credibility intervals, obtained by the quadratures.

Software reliability data

The Weibull software reliability model is used here for modeling the times between 136 successive computer software failures of "SYS1" data (Chang and Liu, 2009) and for estimating the mean time. A very small constant is added to failure times in order to avoid the case of a zero value in the likelihood function, but the value of the constant did not change the estimation results.

The state initial condition is $\lambda_0|Y_0 \sim \text{Gamma}(0.01, 0.01)$ in a similar way to the previous examples. The arguments of the 'ngssm.mle' that should be set are the formula `Ytm~Xtm`, `data.frame(Ytm,Xtm)`, and `model = "SRWeibull"`. An initial static parameter vector is optional (`StaPar = c(0.9,0.7,0.01)`). The maximum likelihood estimation for the "SYS1" data is performed using the R code below.

```
> library(NGSSEML)
> data(sys1_data)
> Ytm = sys1_data[,1]+0.00001
> Xtm = sys1_data[,2]
> Zt = NULL
> model = "SRWeibull"
## LabelParTheta = c("w", "alpha", "Beta1")
> StaPar = c(0.9, 0.7, 0.01)
> fit = ngssm.mle(Ytm ~ Xtm, data = data.frame(Ytm,Xtm), model = model,
+ StaPar = StaPar)
```

Bayesian estimation for the "SYS1" data is obtained via quadrature rules with a grid of 15 points (`pointss=15`) and a sample of 1000 draws of the marginal posterior distribution (`nsamplex=1000`). As in the chassical fit, the following arguments have to be inserted in the 'ngssm.bayes' function: `Ytm~Xtm`, `data = data.frame(Ytm,Xtm)`, and `model = "SRWeibull"`. An initial static parameter vector is optional (`StaPar = c(0.98,0.75,0.02)`. The marginal priors for the static parameter vector $\boldsymbol{\theta} = (w, \nu, \beta_1)$ are $w \sim \text{Beta}(1, 1)$, $\nu \sim \text{Gamma}(0.01, 0.01)$ and $\beta_1 \sim N(0, 100)$ (vague priors) that are stated in the code via the arguments `prw = c(1,1)`, `prnu = c(0.1,0.1)`, `prbetamu = c(0)`, and `prbetasigma = diag(100,1,1)`, respectively.

```
> library(NGSSEML)
> data(sys1_data)
> Ytm = sys1_data[,1]+0.00001
> Xtm = sys1_data[,2]
> model = "SRWeibull"
## LabelParTheta = c("w", "nu", "Beta")
> StaPar = c(0.98,0.75,0.02)
## points
> pointss = 15
## Bayesian Fit:
> fitbayes = ngssm.bayes(Ytm ~ Xtm, data = data.frame(Ytm,Xtm), model = model,
+ pz = NULL, StaPar = StaPar, prw = c(1, 1), prnu = c(0.1, 0.1), prbetamu = c(0),
+ prbetasigma = diag(100, 1, 1), pointss = pointss, nsamplex = 1000, postplot = TRUE,
+ contourplot = TRUE, verbose = TRUE)
```

Table 7 presents the point and interval estimates for the Weibull SR model fitted to the "SYS1" data under the classical and Bayesian approaches. The estimate of parameter w is high (larger than 0.9), which means that the information loss over time is small. Besides, we have strong evidence from the data in favor of positive values for the regression coefficient β_1 and values lower than 1 for the shape parameter ν (i.e., decreasing failure rates).

$\boldsymbol{\theta}$	**MLE**	**95% Conf. Int.**	**Posterior Mean**	**95% Cred. Int.**
w	0.999	[0.996; 1.000]	0.969	[0.895; 0.994]
ν	0.753	[0.648; 0.857]	0.754	[0.655; 0.856]
β_1	0.023	[0.018; 0.029]	0.023	[0.015; 0.031]

Table 7: Point and interval (level 95%) estimation for the Weibull SR model fitted to the "SYS1" data.

The commands `postplot = TRUE` and `contourplot = TRUE` in the code of the Bayesian approach provide Figure 7 that shows the marginal posterior distributions and contour plot of the joint posterior distribution of the parameters w, ν, and β_1, respectively. Parameter w controls the information loss over time, while ν is the shape parameter of the Weibull SR model. The estimate of ν is lower than one, which means strong evidence in favor of decreasing failure rates. Analyzing the marginal posterior of β_1, we can observe that the number of previous failures seems to be relevant to the model.

The arguments `posts = posts`, `Proc = "Smooth"`, `Type = "Marg"`, `transf = 1/4`, and `model = "SRWeibull"` establish a sample of the marginal posterior of the static parameters, the smoothed distribution for the states, integrating the static parameters out, a transformation of the fourth root in the series, and the Weibull SR model are included in the code below to build Figure 8. Figure 8 provides transformed smoothed estimates for the mean of the data under the Weibull SR model. A transformation to reduce the scale of the data was necessary in order to obtain a better visualization of the graph. The fourth root was a suitable transformation in this case. As can be seen in Figure 8, the estimates follow well the behavior of the data.

```
> library(NGSSEML)
## Smoothing
> posts = fitbayes$samplepost
> set.seed(1000)
> PlotF(Ytm ~ Xtm, data = data.frame(Ytm, Xtm), plotYt = TRUE, transf = 1/4,
+ model = "SRWeibull", Proc = "Smooth", Type = "Marg", posts = posts, typeline = 'l',
+ col = c("black", "blue", "lightgrey"), xlab = "Number of Failures",
+ ylab = "Transformed Times", xlim = c(0, 139), ylim = c(-2, 10), lty = c(1, 2, 1),
+ lwd = c(1, 2, 1), cex = 0.68)
```

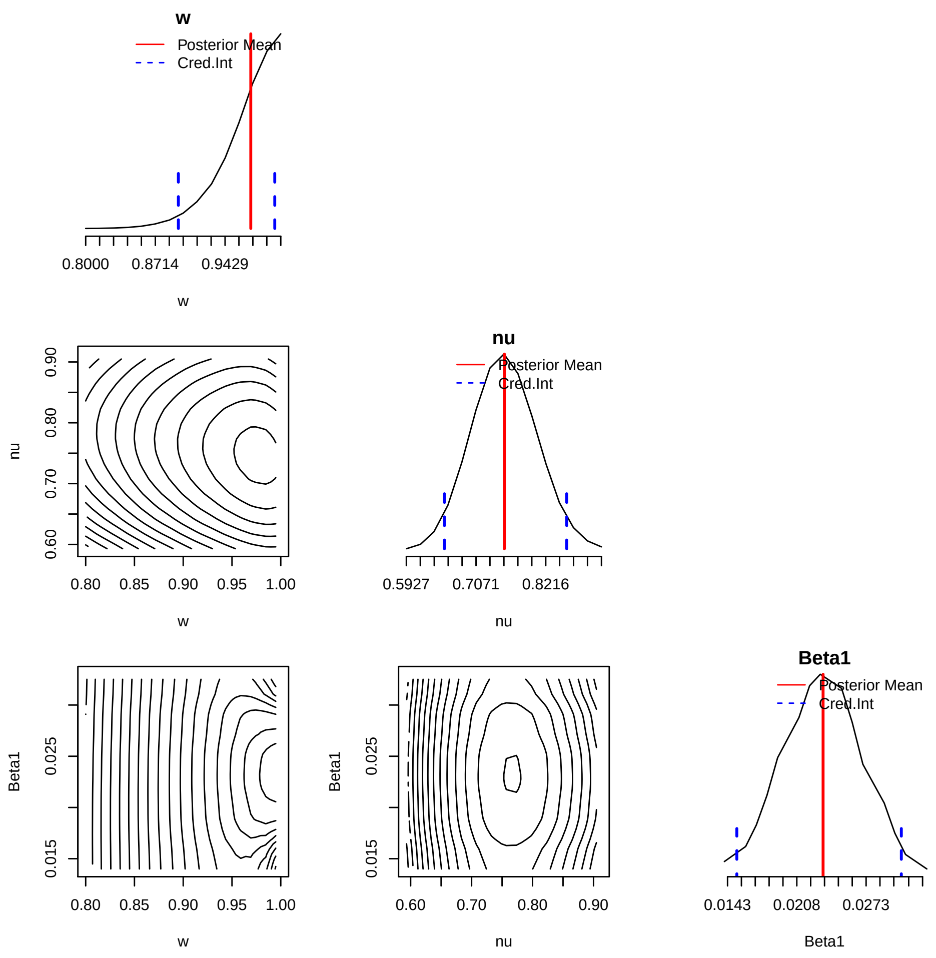


Figure 7: The full and dashed lines indicate, respectively, the fourth root of the series and its smoothed mean, obtained by the fit of the Weibull SR under the Bayesian perspective. The grey area indicates the 95% credibility intervals, obtained by the quadratures.

Conclusion

The package demonstrates how non-Gaussian state-space models (with exact marginal likelihood) can suitably be applied and employed in non-Gaussian time series and reliability analysis using R. The main contribution of the R-package NGSSEML is to provide an easy and fast code for classical and Bayesian estimations in non-Gaussian state space models with the exact marginal likelihood in the R programming language. Four important examples were presented for modeling time series and reliability data. The main advantages of the employed methodology are its analytical and

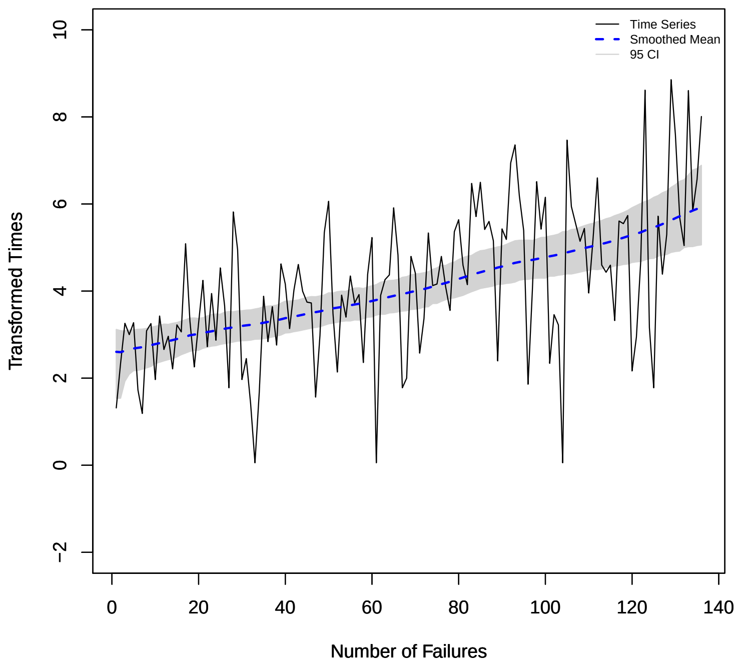


Figure 8: The full and dashed lines indicate, respectively, the fourth root of the series of the times between successive failures and the fourth root of the smoothed mean of the data, obtained by the fit of the SRWeibull under the Bayesian perspective using the quadratures. The grey area indicates the 95% credibility intervals.

computational simplicity, fast routines, and the ability to accommodate different types of distributions, even those which do not belong to the exponential family, combined with exact inference. Both classical and Bayesian approaches may be performed since the exact marginal likelihood for the static parameters is available. The package also provides R codes or functions for the prediction, filtering, and smoothing procedures of the latent states. Also, it is a platform where frequently used models, like the Normal and Poisson, can be included, as well as other volatility and reliability models, like the GED and Generalized Gamma.

Future work aims to increase the number of distributions (special cases) that can be specified for the observations in the package (mainly for time series data), to include dynamic linear models with the mean and variance varying over time that can conditionally be written in the NGSSEML form (Gamerman et al., 2013; Rego and dos Santos, 2020), and to introduce models for modeling multivariate time series data (Aktekin et al., 2018, 2020).

Acknowledgements

T. R. Santos was supported by CNPq-Brazil and FAPEMIG Foundation/Grant 25909. G. C. Franco was supported by CNPq-Brazil and FAPEMIG Foundation. D. Gamerman was supported by CNPq-Brazil and FAPERJ-Brazil. We thank the editor and reviewers for their constructive critique and positive review of our paper.

Bibliography

T. Aktekin, R. Soyer, and F. Xu. Assessment of mortgage default risk via bayesian state space models. *The Annals of Applied Statistics*, 7(3):1450–1473, Apr. 2013. URL https://www.jstor.org/stable/23566480. [p208]

T. Aktekin, N. Polson, and R. Soyer. Sequential bayesian analysis of multivariate count data. *Bayesian Analysis*, 13(2):385–409, June 2018. URL https://doi.org/10.1214/17-BA1054. [p208, 225]

T. Aktekin, N. Polson, and R. Soyer. A family of multivariate non-gaussian time series models. *Journal of Time Series Analysis*, 41(5):385–409, May 2020. URL https://doi.org/10.1111/jtsa.12529. [p225]

C. Andrieu and A. Doucet. Particle filtering for partially observed gaussian state space models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 64:827–836, Oct. 2002. URL https://doi.org/10.1111/1467-9868.00363. [p208]

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer text in Statistics, New York, 1996. URL https://doi.org/10.1007/978-1-4757-2526-1. [p210]

C. Carvalho, M. Johannes, H. Lopes, and N. Polson. Particle learning and smoothing. *Statistical Science*, 25:88–106, Feb. 2010. URL https://doi.org/10.1214/10-STS325. [p208]

Y. Chang and C. Liu. A generalized jm model with applications to imperfect debugging in software reliability. *Applied Mathematical Modelling*, 33:3578–3588, Sept. 2009. URL https://doi.org/10.1016/j.apm.2008.11.018. [p222]

B. Christoffersen, A. Miller, A. Williams, B. developers, and R-core. *dynamichazard: Dynamic Hazard Models using State Space Models*. R-CRAN, Vienna, Austria, 2021. URL https://cran.r-project.org/web/packages/dynamichazard/index.html. [p208]

R. Davis and R. Wu. A negative binomial model for time series of counts. *Biometrika*, 96(3):735–749, Sept. 2009. URL https://doi.org/10.1093/biomet/asp029. [p215]

F. de Pinho, G. Franco, and R. Silva. Modeling volatility using state space models with heavy tailed distributions. *Mathematics and Computers in Simulation*, 139:108–127, Jan. 2016. URL https://doi.org/10.1016/j.matcom.2015.08.005. [p208, 209]

C. Dethlefsen and S. Lundbye-Christensen. Formulating state space models in r with focus on longitudinal regression models. *Journal of Statistical Software*, 16:1–15, Apr. 2006. URL https://doi.org/10.18637/jss.v016.i01. [p208]

J. Doornik. Ox: An object-oriented matrix language. *The Economic Journal*, 107(440):256–259, Jan. 1997. URL https://doi.org/10.1093/ej/107.440.256. [p208]

J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, Oxford, Dec. 2012. URL https://doi.org/10.1093/acprof:oso/9780199641178.001.0001. [p208]

D. Gamerman, T. R. Santos, and G. C. Franco. A non-gaussian family of state-space models with exact marginal likelihood. *Journal of Time Series Analysis*, 34:625–645, Oct. 2013. URL https://doi.org/10.1111/jtsa.12039. [p208, 209, 210, 211, 213, 225]

J. Helske. Kfas: Exponential family state space models in r. *arXiv*, arXiv:1612.01907, 2016. [p208]

J. Helske and M. Vihola. *bssm: Bayesian Inference of Non-Linear and Non-Gaussian State Space Models*. R-CRAN, Vienna, Austria, 2021. URL https://cran.r-project.org/web/packages/bssm/index.html. [p208]

E. Holmes, E. Ward, M. Scheuerell, and K. Wills. *MARSS: Multivariate autoregressive state-space modeling*. R-CRAN, Vienna, Austria, 2013. URL https://cran.r-project.org/web/packages/MARSS/index.html. [p208]

J. Kim and R. Proschan. Piecewise exponential estimator of survivor function. *IEEE Transactions on Reliability*, 40:134–139, June 1991. URL https://doi.org/10.1109/24.87112. [p220]

A. King, D. Nguyen, and E. L. Ionides. Statistical inference for partially observed markov processes via the r package pomp. *Journal of Statistical Software*, 69(12):1–43, Mar. 2016. URL https://doi.org/10.18637/jss.v069.i12. [p208]

S. Koopman, N. Shephard, and J. Doornik. Statistical algorithms for models in state space using ssfpack 2.2. *The Econometrics Journal*, 2(1):107–160, June 1999. URL https://doi.org/10.1111/1368-423X.00023. [p208]

G. Petris. An r package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, Oct. 2010. URL https://doi.org/10.18637/jss.v036.i12. [p208, 209, 210]

A. T. Rego and T. R. dos Santos. Non-gaussian stochastic volatility model with jumps via gibbs sampler. *Statistics and Its Interface*, 13:209–219, July 2020. URL https://dx.doi.org/10.4310/SII.2020.v13.n2.a6. [p225]

B. D. Ripley. Time series in r 1.5.0. *The Newsletter of the R Project*, 2(2):2–7, 2002. [p208]

T. Santos, D. Gamerman, and G. Franco. Reliability analysis via non-gaussian state-space models. *IEEE Transactions on Reliability*, 66:309–318, Mar. 2017. URL https://doi.org/10.1109/TR.2017.2670142. [p208, 209, 210, 213]

D. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, July 1970. URL https://doi.org/10.1090/S0025-5718-1970-0274029-X. [p210]

R. Smith and J. Miller. A non-gaussian state space model and application to prediction of records. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(1):79–88, Sept. 1986. URL https://doi.org/10.1111/j.2517-6161.1986.tb01392.x. [p209]

C. Szymanski. *Ldlmodeler: Generalized Dynamic Linear Modeler*. R-CRAN, Vienna, Austria, 2014. URL http://CRAN.R-project.org/package=dlmodeler. [p208]

R. C. Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p208]

M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer, New York, May 1997. URL https://doi.org/10.1007/b98971. [p209]

M. West, P. Harrison, and H. Migon. Dynamic generalized linear models and bayesian forecasting (with discussion). *Journal of the American Statistical Association*, 73–97:80, Mar. 1985. URL https://doi.org/10.1080/01621459.1985.10477131. [p208]

S. Zeger. A regression model for time series of counts. *Biometrika*, 75:621–629, Dec. 1988. URL https://doi.org/10.2307/2336303. [p213]

D. Zes. *SSsimple: State Space Models*. R-CRAN, Vienna, Austria, 2019. URL https://cran.r-project.org/web/packages/SSsimple/index.html. [p208]

Thiago R. Santos
Universidade Federal de Minas Gerais
Brazil
ORCID: 0000-0003-2244-2718
thiagords@est.ufmg.br

Dani Gamerman
Universidade Federal do Rio de Janeiro
Brazil
dani@im.ufrj.br

Glaura C. Franco
Universidade Federal de Minas Gerais
Brazil
glaura@est.ufmg.br

BayesSenMC: an R package for Bayesian Sensitivity Analysis of Misclassification

by Jinhui Yang, Lifeng Lin and Haitao Chu

Abstract In case–control studies, the odds ratio is commonly used to summarize the association between a binary exposure and a dichotomous outcome. However, exposure misclassification frequently appears in case–control studies due to inaccurate data reporting, which can produce bias in measures of association. In this article, we implement a Bayesian sensitivity analysis of misclassification to provide a full posterior inference on the corrected odds ratio under both non-differential and differential misclassification. We present an R (R Core Team, 2018) package **BayesSenMC**, which provides user-friendly functions for its implementation. The usage is illustrated by a real data analysis on the association between bipolar disorder and rheumatoid arthritis.

Introduction

Many epidemiological studies are concerned with assessing the risk of an outcome between exposed and non-exposed subjects. For example, in a case–control study, researchers first identify subjects who have the disease of interest (the case group) and subjects who do not (the control group), and then ascertain the exposure status of the subjects in each group. The odds ratio is typically used to assess the association between the exposure and disease in the case–control study; it describes the ratio of the exposure odds in the case group to that in the control group.

However, misclassification of exposure, disease outcome, or covariates appears frequently in observational studies of epidemiological or medical research (Rothman et al., 2008; Brakenhoff et al., 2018). In a case–control study, misclassification is often due to inaccurate reporting of the exposure status (e.g., self-reported data). This can consequently lead to biased estimation of exposure probabilities and odds ratio. To adjust for such biases, we can correct the odds ratio using the observed data from the case–control study and the sensitivity and specificity of correctly classifying exposure status from external data. Here, the sensitivity is the proportion of exposed subjects that are correctly classified as exposed (i.e., true positive), and the specificity is the proportion of non-exposed subjects that are correctly classified as non-exposed (i.e., true negative).

Quantitative assessment of misclassification bias is necessary to estimate uncertainty in study results. There are many statistical methods for misclassification correction; nearly all of them use prior information that maps observed measurements to true values (Greenland, 2005). These methods include regression calibration and multiple imputation (Rosner et al., 1989; Spiegelman et al., 2001; Cole et al., 2006), in which the mapping is based on a validation study. Also, sensitivity analysis can be used to evaluate the effects of uncertainties in measurement on the observed results of the study (Greenland, 1996; Lash and Flink, 2003; Chu et al., 2006), in which the mapping from observed to true measurements may be based on prior information or expert opinion about the accuracy of the measurement. However, when such information or opinion is lacking, researchers may over- or under-adjust for misclassification with an inaccurate guess, which may, in turn, produce a poor estimate (Gustafson et al., 2006).

Moreover, despite the ubiquity of measurement error, these methods remain rarely used due to the complexity of statistical approaches, especially the complexity of prior specifications as well as the lack of software packages (Lash and Flink, 2003). For example, in a random sample survey of 57 epidemiological studies (Jurek et al., 2006), only one study used quantitative corrections. Sensitivity analysis is simple but limited insofar as it does not provide formal interval estimates that combine uncertainty due to random error with misclassification. Several authors have addressed this deficiency by using probabilistic (Monte Carlo) sensitivity analyses (Greenland, 2005). For example, Fox et al. (2005) proposed a probabilistic sensitivity analysis of misclassified binary variables based on multiple imputation; they provided SAS code and Excel macro for this approach. Such methods can be viewed as means of summarizing the bias over sensitivity analyses using a prior distribution about the bias parameters (Greenland, 2005). Many of them have been implemented in the R package **episensr** (Haine, 2021). Specifically, **episensr** allows for specifications of prior distributions for sensitivity and specificity, such as uniform and logit normal, as well as sequential bias modeling that can be applied to more than one type of bias, such as for both misclassification and selection biases.

Other methods employ a Bayesian implementation of the probabilistic bias analysis or perform an outright Bayesian analysis (Greenland, 2005; Chu et al., 2006; MacLehose and Gustafson, 2012; Gustafson et al., 2006). The Bayesian analysis is substantially different from conventional probabilistic sensitivity analysis and much more flexible for incorporating different types of priors. However, it is often computationally expensive and more difficult to conduct by general users without a statistical background. Gustafson et al. (2006) accounted for the prior uncertainties of sensitivity and specificity in the evaluation of the results of a case–control study. MacLehose and Gustafson (2012) compared a Bayesian approach with probabilistic bias analysis based on a case–control study of congenital defects, concluding that the two approaches are mostly similar if using similar prior data admissibility as well as uniform priors on exposure probabilities.

This article focuses on an R package correcting for exposure misclassification in a case–control study. Extending from the Bayesian approach introduced by Gustafson et al. (2006), we implement the methods outlined in Chu et al. (2006), which account for the correlation between the sensitivity and specificity in the model specification. The methods can be applied to both non-differential and differential misclassification; that is, the degree of misclassification can be the same across the case and control groups or distinctly different. Furthermore, we use the generalized linear mixed bivariate effects model introduced by Chu et al. (2010) to jointly model the sensitivity and specificity that may be informed by an external meta-analysis on the diagnostic accuracy of the exposure factor.

This article introduces the implementation of the methods for misclassification via our R package **BayesSenMC** (Bayesian sensitivity analysis by Monte Carlo sampling). The package is mainly implemented in Stan, an imperative probabilistic programming language, which uses Hamiltonian Monte Carlo (HMC), a form of efficient Markov Chain Monte Carlo (MCMC) sampling.

An illustrative example

This section presents an illustrative case–control study on the association between bipolar disorder and rheumatoid arthritis, originally investigated by Farhi et al. (2016); this example will also be used to demonstrate the implementation of the methods for misclassification. The exposure is bipolar disorder, and the disease outcome is rheumatoid arthritis, which is a chronic autoimmune disorder that primarily affects joints and occurs in nearly 1% of the population in developed countries (McInnes and Schett, 2017). Table 1 presents the data.

	Bipolar Disorder		
Rheumatoid arthritis	Exposed	Unexposed	Total
Case	66	11,716	11,782
Control	243	57,730	57,973

Table 1: Counts of the case–control study of the association between bipolar disorder and rheumatoid arthritis.

The unadjusted odds ratio is 1.34 with the 95% confidence interval (CI) (1.02, 1.76), indicating a significant association between rheumatoid arthritis and bipolar disorder. Of note, Farhi et al. (2016) acknowledged the limitation that “lack of validation of the diagnosis of bipolar disorder in the subjects cannot be completely excluded.” For example, bipolar disorder can be classified as type I, type II, etc.; it is especially difficult to diagnose bipolar disorder type II (Phillips and Kupfer, 2013).

Assuming certain fixed values or prior distributions of sensitivity and specificity, we can use the method by Chu et al. (2006) to correct the odds ratio accounting for the exposure misclassification. The sensitivity and specificity can be either some fixed values or random variables following some prior distributions.

The prior distributions can be estimated from external evidence using a meta-analysis, e.g., using the bivariate generalized linear mixed model approach proposed by Chu et al. (2010). The following section presents the details of these methods. This article uses the meta-analysis performed by Carvalho et al. (2015) to obtain the prior distributions of the sensitivity and specificity of classifying the exposure status of bipolar disorder. The meta-analysis contains three subgroups of screening detection instruments: bipolar spectrum diagnostic scale (8 studies with sensitivity between 0.52 and 0.90 and specificity between 0.51 and 0.97), hypomania checklist (17 studies with sensitivity between 0.69 and 1.00 and specificity between 0.36 and 0.98), and mood disorder questionnaire (30 studies with sensitivity between 0.00 and 0.91 and specificity between 0.47 and 1.00). The dataset from the Clalit Health Services (the largest Health Maintenance Organization in Israel) used in Farhi et al. (2016) does not specify the exact screening detection instrument for identifying the bipolar disorder. Therefore, we will use all three subgroups' data (55 studies in total) in Carvalho et al. (2015) for our analysis on the sensitivity and specificity. A subset of the meta-analysis data is shown in Table 2.

According to their definitions, the study-specific sensitivity and specificity can be estimated as (the number of true positives) / (the number of true positives plus the number of false negatives) and

Study ID	True positive	False negative	True negative	False positive
1	81	9	444	427
2	12	3	44	19
3	74	26	97	3
4	52	16	23	4
5	228	113	18	4
⋮	⋮	⋮	⋮	⋮
55	63	6	32	13

Table 2: The meta-analysis on diagnosis accuracy of bipolar disorder performed by Carvalho et al. (2015).

(the number of true negatives) / (the number of true negatives plus the number of false positives). For example, study 1 gives the sensitivity 81/(81 + 9) = 0.90 and the specificity 444/(444 + 427) $\approx$ 0.51. The generalized linear mixed-effects model will be used to synthesize all 55 studies to estimate the overall sensitivity and specificity.

Methods

In this section, we introduce the specific models and methods to deal with misclassification.

Bayesian approach to correcting misclassification bias

Consider a case–control study, and we are interested in the odds ratio from this study. Table 3 presents the notation of the observed data. The odds ratio is estimated as

$$\widehat{\mathrm{OR}} = \frac{ad}{bc}.$$

When the odds ratio is larger or smaller than 1, the exposure happens more or less likely in the case group, suggesting an association between the disease status and the exposure status. On the other hand, the odds ratio close to 1 suggests that the disease and the exposure are less likely associated.

Group	Exposed	Unexposed	Total
Case	a	b	N_1
Control	c	d	N_0

Table 3: Observed counts of a case–control study.

Assume that the observed exposure probability is P_k, the true exposure probability is π_k, the sensitivity is Se_k, and the specificity is Sp_k for group k (k = 1 for the case group and 0 for the control group) in the case–control study. Then, we can represent the observed exposure probability in terms of the true exposure probability, the sensitivity, and the specificity:

$$\begin{aligned} P_k &= \mathsf{P}(\text{observed } E \text{ in group } k) \\ &= \mathsf{P}(\text{observed } E \mid \text{true } E \text{ in group } k)\mathsf{P}(\text{true } E \text{ in group } k) \\ &\quad + \mathsf{P}(\text{observed } E \mid \text{true } \overline{E} \text{ in group } k)\mathsf{P}(\text{true } \overline{E} \text{ in group } k) \\ &= Se_k\pi_k + (1 - Sp_k)(1 - \pi_k), \end{aligned} \tag{1}$$

where E denotes exposure and $\overline{E}$ denotes non-exposure. This yields

$$\pi_k = (P_K + Sp_k - 1)/(Se_k + Sp_k - 1).$$

Consequently, the misclassification-corrected odds ratio can be calculated as

$$\mathrm{OR_c} = \frac{\pi_1/(1-\pi_1)}{\pi_0/(1-\pi_0)} = \frac{(P_1 + Sp_1 - 1)(Se_0 - P_0)}{(P_0 + Sp_0 - 1)(Se_1 - P_1)}. \tag{2}$$

Based on Equation (1), we can specify the following Bayesian hierarchical model to estimate the corrected odds ratio of the case–control study, with a and c being the observed counts from Table 3:

$$\begin{aligned} \text{Likelihood:}\quad & a \sim Bin(N_1, P_1) \text{ and } c \sim Bin(N_0, P_0); \\ \text{Link:}\quad & P_k = Se_k \pi_k + (1 - Sp_k)(1 - \pi_k), \quad k = 0, 1; \\ & \text{LOR}_\text{c} = \text{logit}(\pi_1) - \text{logit}(\pi_0) \text{ and } \text{OR}_\text{c} = \exp(\text{LOR}_\text{c}); \\ \text{Prior:}\quad & \text{logit}(\pi_0) \sim N(0, 10^2) \text{ and } \text{LOR}_\text{c} \sim N(0, 2^2); \\ & Se_k, Sp_k \sim f(\cdot). \end{aligned} \tag{3}$$

Here, we assume weakly-informative priors for the true exposure probabilities π_0 and π_1, which give a 95% CI of the true odds ratio between $e^{-2\times1.96}$ (≈ 0.02) and $e^{2\times1.96}$ (≈ 50.40), and a binomial distribution for the number of observed exposure in case and control studies. Thus, the Bayesian inference can be formulated as a posterior distribution of π_0, π_1, and the corrected odds ratio.

Additionally, the function $f(\cdot)$ denotes the joint prior for the sensitivity and specificity (for either non-differential or different misclassification). In practice, the sensitivity and specificity are not available from the case–control study, and they may be estimated as certain fixed values by subjective experts' opinions.

Alternatively, we may consider incorporating evidence-based prior information from existing studies on the diagnostic accuracy of the exposure status (e.g., the data in Table 2). This allows us to account for uncertainties in the sensitivity and specificity and potential correlation between them. The next subsection presents methods to obtain the prior information for the sensitivity and specificity from a meta-analysis.

Estimating prior distributions on the sensitivity and specificity from a meta-analysis

This section briefly discusses the generalized linear mixed-effects model (GLMM) to estimate priors on the sensitivity and specificity. Suppose that a meta-analysis on the diagnostic accuracy of the exposure status is available as external data to inform the priors of sensitivity and specificity that are needed to correct the odds ratio in the case–control study. Denote the number of independent studies in the meta-analysis by m, and let n_{i11}, n_{i00}, n_{i01}, and n_{i10} be the number of true positives, true negatives, false positives, and false negatives, respectively, in study i ($i = 1, \ldots, m$). Consequently, there are $n_{i11} + n_{i10}$ truly exposed subjects and $n_{i00} + n_{i01}$ truly unexposed subjects.

Assuming that n_{i11} and n_{i00} follow binomial distributions given the number of exposed and unexposed subjects, respectively, the bivariate GLMM can be specified as (Chu et al., 2010; Ma et al., 2016):

$$\begin{aligned} & n_{i11} \sim Bin(n_{i11} + n_{i10}, Se_i) \text{ and } n_{i00} \sim Bin(n_{i00} + n_{i01}, Sp_i), \quad i = 1, \ldots, m; \\ & g\left(\frac{Se_i - Se^L}{Se^U - Se^L}\right) = u + \mu_i \text{ and } g\left(\frac{Sp_i - Sp^L}{Sp^U - Sp^L}\right) = v + \nu_i; \\ & \begin{bmatrix} \mu_i \\ \nu_i \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\mu^2 & \rho\sigma_\mu\sigma_\nu \\ \rho\sigma_\mu\sigma_\nu & \sigma_\nu^2 \end{bmatrix}\right), \end{aligned} \tag{4}$$

where u and v are the fixed effects implying the overall sensitivity and specificity in all m studies, and μ_i and ν_i are the study-specific random effects. Also, σ_μ^2 and σ_ν^2 describe the heterogeneity of the underlying sensitivity and specificity across studies, and ρ models the correlation between the sensitivity and specificity. We denote the estimated fixed effects by $\hat{u}$ and $\hat{v}$, the estimated variances as $\hat{\sigma}_\mu^2$ and $\hat{\sigma}_\nu^2$, and the estimated correlation coefficient as $\hat{\rho}$.

The lower and upper bounds Se^L, Se^U, Sp^L, and Sp^U provide constraints on sensitivity and specificity, which are chosen to exclude all improbable values. A smaller difference between Se^L and Se^U (or between Sp^L and Sp^U) indicates higher confidence in the diagnostic accuracy of the exposure status. When there is no confidence for the range, we can set $Se^L = Sp^L = 0$ and $Se^U = Sp^U = 1$. Alternatively, setting $Se^L = Sp^L = 0.5$ indicates that the diagnosis of exposure is better than chance. For simplicity of implementation, we only allow the same lower and upper bounds for Se and for Sp in our package **BayesSenMC**. In addition, $g(\cdot)$ is the link function (e.g., the logit, probit, and complementary log-log). The logit link, $\text{logit}(t) = \log \frac{t}{1-t}$, is commonly used in practice, and our package **BayesSenMC** adopts this link.

Recall that the Bayesian hierarchical model for estimating the corrected odds ratio in the case–control study in Equation (3) specifies a joint prior $f(\cdot)$ for the sensitivity and specificity. We consider six specifications for this prior as follows:

(i) Crude (uncorrected) odds ratio: no misclassification. The specification of the prior is equivalent to setting $Se_0 = Se_1 = Sp_0 = Sp_1 = 1$. Consequently, $P_1 = \pi_1$ and $P_0 = \pi_0$.

(ii) Corrected OR: misclassification of the exposure status exists, and the sensitivity and specificity for both cases and controls are assumed to be fixed values. These fixed values can be directly plugged in the Bayesian model in Equation (3).

(iii) Logit-prior corrected OR: non-differential misclassification of the exposure status exists ($Se_0 = Se_1 = Se$ and $Sp_0 = Sp_1 = Sp$), and the uncertainties of the sensitivity and specificity are considered independently by using normal priors on the logit scale. The evidence of the priors comes from the diagnostic meta-analysis performed in the GLMM in Equation (4). Specifically, we can assign

$$\operatorname{logit}\left(\frac{Se - Se^L}{Se^U - Se^L}\right) \sim N(\hat{u}, \hat{\sigma}_\mu^2) \text{ and } \operatorname{logit}\left(\frac{Sp - Sp^L}{Sp^U - Sp^L}\right) \sim N(\hat{v}, \hat{\sigma}_v^2)$$

as the priors in the Bayesian hierarchical model for the case–control study in Equation (3).

(iv) Fixed-correlation corrected OR: non-differential misclassification of the exposure status exists ($Se_0 = Se_1 = Se$ and $Sp_0 = Sp_1 = Sp$), and the sensitivity and specificity have a joint normal prior on the logit scale to account for their correlation. In practice, the sensitivity is very likely correlated with the specificity when dichotomizing a continuous measurement (Chu and Cole, 2006). Specifically, we use the following bivariate joint prior

$$\begin{bmatrix} \operatorname{logit}\left(\frac{Se-Se^L}{Se^U-Se^L}\right) \\ \operatorname{logit}\left(\frac{Sp-Sp^L}{Sp^U-Sp^L}\right) \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}, \begin{bmatrix} \hat{\sigma}_\mu^2 & \hat{\rho}\hat{\sigma}_\mu\hat{\sigma}_v \\ \hat{\rho}\hat{\sigma}_\mu\hat{\sigma}_v & \hat{\sigma}_v^2 \end{bmatrix}\right).$$

Compared with the previous prior specification with independent sensitivity and specificity, the correlation coefficient $\hat{\rho}$ is additionally considered here. It is also estimated from the GLMM in Equation (4).

(v) Random-correlation corrected OR: in addition to the above bivariate joint prior for the non-differential sensitivity and specificity, we can also consider modeling the uncertainties in the estimated correlation coefficient. We consider applying Fisher's z-transformation to the correlation coefficient in the GLMM. Specifically, instead of directly estimating the correlation coefficient ρ in Equation (4), we reparameterize $\rho = \frac{\exp(2z)-1}{\exp(2z)+1}$ and obtain the point estimate of z from the GLMM and its standard error, denoted by $\hat{z}$ and s_z, respectively. These estimates can be subsequently used as the priors for the sensitivity and specificity in the case–control study:

$$\begin{bmatrix} \operatorname{logit}\left(\frac{Se-Se^L}{Se^U-Se^L}\right) \\ \operatorname{logit}\left(\frac{Sp-Sp^L}{Sp^U-Sp^L}\right) \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}, \begin{bmatrix} \hat{\sigma}_\mu^2 & \rho\hat{\sigma}_\mu\hat{\sigma}_v \\ \rho\hat{\sigma}_\mu\hat{\sigma}_v & \hat{\sigma}_v^2 \end{bmatrix}\right);$$
$$\rho = \frac{\exp(2z)-1}{\exp(2z)+1};$$
$$z \sim N(\hat{z}, s_z^2).$$

(vi) Differential corrected OR: finally, we consider the differential misclassification of the exposure status, i.e., $Se_0 \neq Se_1$ and $Sp_0 \neq Sp_1$. All above choices of priors can be similarly applied to the four-variate set $\{Se_0, Sp_0, Se_1, Sp_1\}$. For simplicity, we consider a joint prior similar to that in (iv). However, the prior applies to cases and controls separately, and it does not account for the uncertainties in the correlation coefficient as in (v). That is,

$$\begin{bmatrix} \operatorname{logit}\left(\frac{Se_k-Se^L}{Se^U-Se^L}\right) \\ \operatorname{logit}\left(\frac{Sp_k-Sp^L}{Sp^U-Sp^L}\right) \end{bmatrix} \sim N\left(\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix}, \begin{bmatrix} \hat{\sigma}_\mu^2 & \hat{\rho}\hat{\sigma}_\mu\hat{\sigma}_v \\ \hat{\rho}\hat{\sigma}_\mu\hat{\sigma}_v & \hat{\sigma}_v^2 \end{bmatrix}\right), \quad k = 0, 1.$$

Because of the complexity of the Bayesian model in Equation (3) with the above various choices of priors for the sensitivity and specificity, we will use Markov chain Monte Carlo (MCMC) sampling to produce the posterior distribution and thus estimate the misclassification-bias-corrected odds ratio in the case–control study and its credible interval.

Implementation in R

The aforementioned methods can be implemented in the R package **BayesSenMC**. The function `nlmeNDiff` fits a non-differential GLMM and returns a **lme4** (Bates et al., 2021) object, for which commands such as `summary` can be used to extract useful statistics from the model; see `methods(class`

= "merMod") for more details. Users can also call the paramEst function to get a list of specific parameter estimates of the fit that can be directly inputted into the model functions of **BayesSenMC** for Bayesian inferences. In addition, the link function used in nlmeNDiff can be modified by specifying lower and upper, which then changes the lower and upper bounds of Se_k and Sp_k ($k = 1$ for cases and 0 for controls).

The package **BayesSenMC** includes six model functions and one graphing function called plotOR. The model functions return an S4 object of type stanfit, an instance of **rstan** (Stan Development Team, 2020), which is an interface of Stan (Carpenter et al., 2017) in R. Users can call methods such as print or extract to get detailed information about the posterior samples. The MCMC procedures are implemented with a default of two chains, each with 1000 iterations of burn-in period and 2000 iterations to estimate the posterior parameters. They are fit using stan, and the default Monte Carlo algorithm is the No-U-Turn sampler, a variant of Hamiltonian Monte Carlo (Hoffman and Gelman, 2014; Betancourt, 2017). Any additional arguments to the model function call will be passed into stan. The returned object can then be inputted into plotOR to visualize the posterior distribution of the adjusted odds ratio, as well as the probability density lines of odds ratio in the cases of no misclassification and constant Se/Sp as comparisons to the posterior distribution. It takes optional argument passed into geom_histogram, and returns a **ggplot2** (Wickham et al., 2021) object that can be further customized.

The latest version of **BayesSenMC** is available from CRAN. The package can be directly installed via the R prompt:

```
R> install.packages("BayesSenMC")
R> library("BayesSenMC")
```

Example in R

In this section, we use the data in Table 1 as well as Table 2 of meta-analysis data on the diagnosis accuracy of bipolar disorder to demonstrate the capabilities of **BayesSenMC**. The analyses are conducted using R version 4.1.0 (2021-05-18).

We first fit the meta-analysis data using the GLMM procedure implemented in our package, assuming non-differential misclassification. Given the range of Se and Sp of the bipolar disorder meta-analysis data, we must only assume $Se^L = Sp^L = 0$ and $Se^U = Sp^U = 1$ for the GLMM to compute real-value results. However, with more information about the type of diagnoses in Farhi et al. (2016), one can find more informative constraints on Se and Sp to fit a more precise model.

```
R> data(bd_meta)
R> my.mod <- nlmeNDiff(bd_meta, lower = 0)
R> my.mod

Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit(0, 1) )
Formula: cbind(Y, N - Y) ~ ((0 + Se + Sp) | sid) + Se
   Data: dat_final
      AIC       BIC    logLik  deviance  df.resid
 851.6825  865.1849 -420.8412  841.6825       105
Random effects:
 Groups Name Std.Dev. Corr
 sid    Se   0.7116
        Sp   0.8935   -0.38
Number of obs: 110, groups:  sid, 55
Fixed Effects:
(Intercept)           Se
    1.12626     -0.05746
```

The indicator variable Se has a value of 1 for Se estimates and 0 for Sp estimates. The random effects are grouped within each study, numbered after sid.

The fit reports the Akaike information criterion (AIC), which can be used to compare across models. The logit means of Se and Sp are given by the fixed effects, 1.069 and 1.126, which translate to a sensitivity of 0.744 and a specificity of 0.755. The values, larger than 0.5, suggest that overall, the diagnostic accuracy for bipolar disorder given our meta-data is better than random, albeit nowhere near perfect. The standard deviations of logit Se and Sp are given by the random effects, as well as the correlation. All of the mentioned parameter estimates can be returned in a list by calling `paramEst`.

We then plug the parameter estimates to get the posterior distributions for the corrected odds ratio given different priors of Se and Sp. We run all 6 different models with the case–control study observations in Farhi et al. (2016), shown in Table 1. The model specifications are shown in the previous subsection.

```
R> params <- paramEst(my.mod)
R> m.1 <- crudeOR(a = 66, N1 = 11782, c = 243, N0 = 57973, chains = 3, iter = 10000)
R> m.2 <- correctedOR(a = 66, N1 = 11782, c = 243, N0 = 57973, prior_list = params,
+      chains = 3, iter = 10000)
R> m.3 <- logitOR(a = 66, N1 = 11782, c = 243, N0 = 57973, prior_list = params,
+      chains = 3, iter = 10000)
R> m.4 <- fixedCorrOR(a = 66, N1 = 11782, c = 243, N0 = 57973, prior_list = params,
+      chains = 3, iter = 10000)
R> m.5 <- randCorrOR(a = 66, N1 = 11782, c = 243, N0 = 57973, prior_list = params,
+      chains = 3, iter = 10000)
R> m.6 <- diffOR(a = 66, N1 = 11782, c = 243, N0 = 57973, mu = c(1.069, 1.069, 1.126, 1.126),
+      s.lg.se0 = 0.712, s.lg.se1 = 0.712, s.lg.sp0 = 0.893, s.lg.sp1 = 0.893,
+      corr.sesp0 = -0.377, corr.sesp1 = -0.377, corr.group = 0, chains = 3,
+      iter = 10000, traceplot = TRUE)

# get summary of model output
R> m.1
```

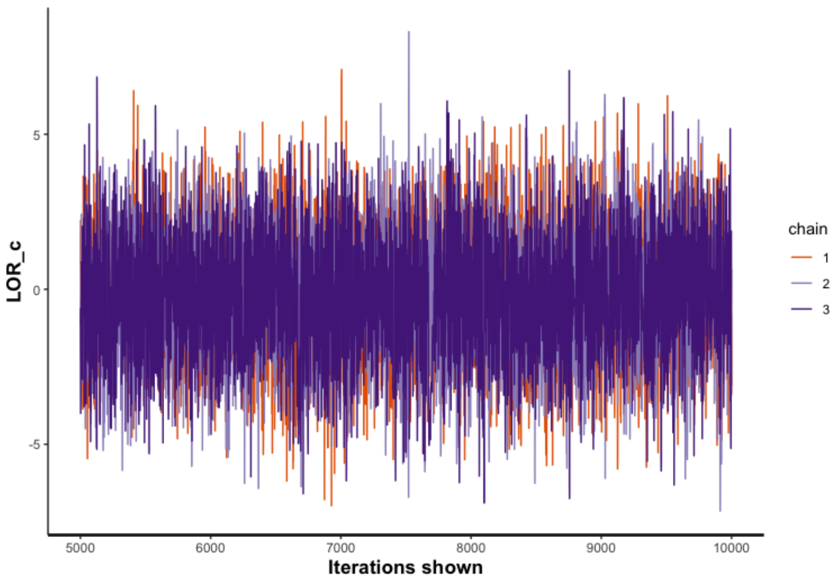


Figure 1: Traceplot of 3 Markov chains with 10,000 iterations for randomly correlated logit bivariate model.

Each model above is implemented with 3 Markov chains, and each chain consists of 5000 burn-in samples and 10,000 iterations to estimate the parameters (Figure 1). The posterior mean, median and 95% confidence limits of the adjusted odds ratio are as below: 1.35 (1.34, 1.01, 1.74), 5.63 (0.85, 0.02, 36.10), 8.61 (1.93, 0.03, 62.27), 9.33 (1.95, 0.03, 62.62), 9.05 (2.00, 0.03, 64.33), 5.23 (0.82, 0.02, 32.64). To obtain and analyze the model output, one can simply call the model variable (e.g., `m.1`). The summary displays the parameters for the model as well as the mean and confidence limits of the adjusted odds ratio (i.e., `ORadj`). One can also specify `traceplot = TRUE` to display a plot of sampled corrected log odds ratio values over iterations, such as in the above `diffOR` method call.

The above example demonstrates the significance of sensitivity and specificity in a case–control study. We can examine that by the ratio of upper to lower 95% posterior interval: 1.74/1.01 = 1.72, 36.10/0.02 = 1805, 62.27/0.03 = 2075.67, 62.62/0.03 = 2087.33, 64.33/0.03 = 2144.33, and 32.64/0.02 = 1632. The greatest jump happens when we assume misclassification in the case–control study, and it only differs slightly with more uncertainties in the model. The increase is especially significant in Farhi et al. (2016) because the estimated mean Se and Sp are around only 0.75, as seen from the GLMM.

In the future, we will consider adding other specifications of priors for sensitivity and specificity to our package, such as beta priors.

```
R> library("ggplot2")
R> g1 <- plotOR(m.1, a = 66, N1 = 11782, c = 243, N0 = 57973, se = 0.744,
+      sp = 0.755, x.max = 3, y.max = 5, binwidth = 0.1) + ggtitle("(i)")
#...... please see supplementary R script for rest of code ......
```

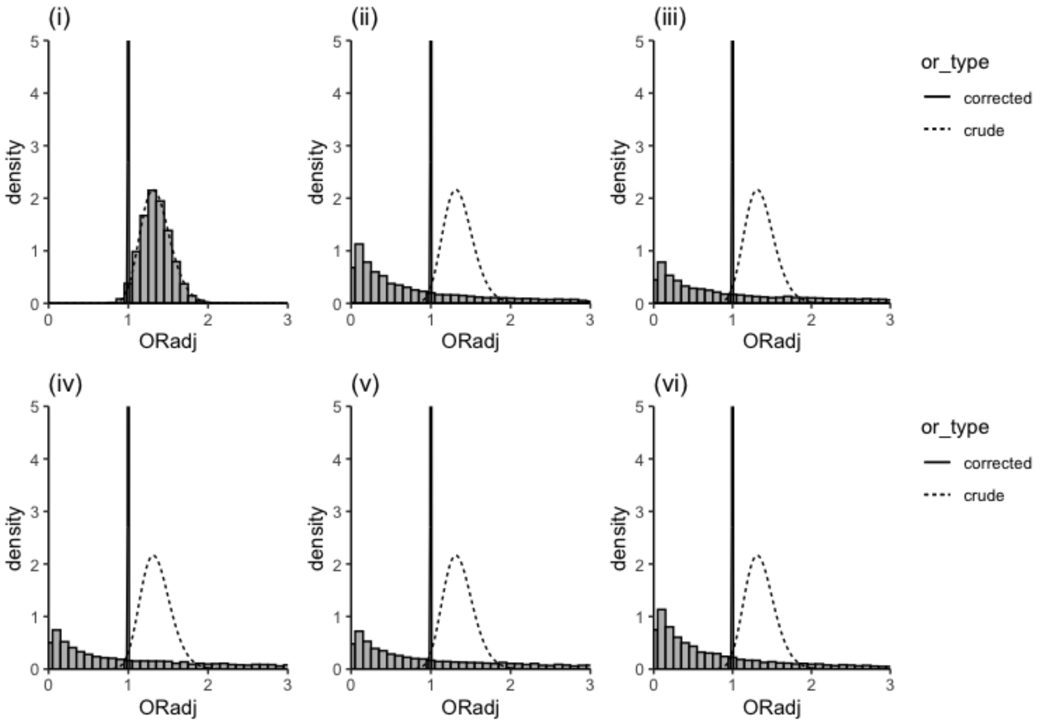


Figure 2: Visualization of posterior distributions of odds ratio for all models. (i) crude (uncorrected) odds ratio with no misclassification; (ii) corrected OR with constant misclassification (Se = 0.744 and Sp = 0.755); (iii) corrected OR with logit bivariate normal misclassification; (iv) corrected OR that extends from (iii) but with constant correlation between Se and Sp; (v) corrected OR that extends from (iii) but with Fisher's z-transformed correlation; (vi) corrected OR with differential misclassification. The dotted and solid lines are the probability density lines of crude OR (i) and corrected OR with no misclassification (ii), respectively, assuming log-normality on odds ratio.

We also implement a graphing function, `plotOR`, which takes the input of a model built with one of the above methods, the observations of the same case–control study, and the estimated Se and Sp from the GLMM. The method visualizes the posterior distribution of that model and plots the probability density line of the adjusted odds ratio given no misclassification (crude OR) and constant misclassification as specified by `Se` and `Sp` (corrected OR). This makes it easy for users to compare the current posterior distribution (especially for models with more uncertainty) with more certain models to visualize the effect of misclassification in a case–control study. In addition, the lines serve as references when comparing across models. The plots and relevant codes are shown in Figure 2. Users can also choose to extract the data from the **rstan** objects by calling functions such as `extract`, `as.data.frame`, etc.

According to the plot, we observe a drastic change to the posterior distribution after taking non-perfect Se and Sp into account. Then, we observe slightly more uniform distributions as there is more uncertainty in the model. What is also worth noting is that in part (ii) of the plot, the posterior density and MCMC sampling do not share the same shape, even though both assume non-perfect constant Se and Sp. This may be a result of low Se and Sp values, which may affect the log-normality assumption in the MCMC posterior samples.

We now show the effects of the number of iterations and chains on the computing speed of our models. All models have been pre-compiled, which reduces the computing time significantly. For example, `randCorrOR`, which is presumably one of the most complex and time-consuming models to compute, takes about 1.32 seconds to run 3 chains with 5000 warm-up periods and 10,000 iterations each. In comparison, it takes about 0.20 seconds to compute 2 chains with 1000 warm-up periods and

2000 iterations each. In practice, a larger number of MCMC chains and iterations leads to more stable and accurate results and thus is recommended. Furthermore, we find that models, such as provided by `randCorrOR`, have smaller target posterior distribution regions in a Markov chain, thus rendering it easy for the algorithm to miss the true distribution and result in "divergent transitions," which may return biased estimates. Increasing the value of `adapt_delta` parameter up to 1 in the `control` argument of the methods can effectively make **rstan** take smaller steps to approach the target.

Conclusion

In this article, we introduce and implement the methods for making posterior inferences on the corrected odds ratio by modeling the uncertainty on both differential and non-differential misclassification through appropriate prior distributions. The specific implementation is publicly available using the `R` package **BayesSenMC**. The process can be divided into two parts. First, one can use the GLMM model with a binomial-logit link to estimate prior information on Se and Sp via a meta-analysis on the misclassification of exposure status. Second, the estimates can be plugged into the modeling functions to provide inferences for the odds ratio. The models can also be visualized side-by-side for better comparisons. The validity of the analyses depends highly on the relevance of meta-analysis, in which irrelevant studies may skew the prior estimates of Se and Sp significantly, and consequentially, the corrected odds ratio. In addition, our models assume normal and independent priors on true exposure probabilities, which may be limited in some cases (Chu et al., 2006).

Bibliography

D. Bates, M. Maechler, B. Bolker, and S. Walker. **lme4**: *Linear Mixed-Effects Models using 'Eigen' and S4*, 2021. URL `https://CRAN.R-project.org/package=lme4`. R package version 1.1-27.1. [p232]

M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017. [p233]

T. B. Brakenhoff, M. Mitroiu, R. H. Keogh, K. G. M. Moons, R. H. H. Groenwold, and M. van Smeden. Measurement error is often neglected in medical literature: a systematic review. *Journal of Clinical Epidemiology*, 98:89–97, 2018. doi: 10.1016/j.jclinepi.2018.02.023. [p228]

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: a probabilistic programming language. *Journal of Statistical Software*, 76(1): 1–32, 2017. doi: 10.18637/jss.v076.i01. [p233]

A. F. Carvalho, Y. Takwoingi, P. M. G. Sales, J. K. Soczynska, C. A. Köhler, T. H. Freitas, J. Quevedo, T. N. Hyphantis, R. S. McIntyre, and E. Vieta. Screening for bipolar spectrum disorders: a comprehensive meta-analysis of accuracy studies. *Journal of Affective Disorders*, 172:337–346, 2015. doi: 10.1016/j.jad.2014.10.024. [p229, 230]

H. Chu and S. R. Cole. Bivariate meta-analysis of sensitivity and specificity with sparse data: a generalized linear mixed model approach. *Journal of Clinical Epidemiology*, 59(12):1331–1332, 2006. doi: 10.1016/j.jclinepi.2006.06.011. [p232]

H. Chu, Z. Wang, S. R. Cole, and S. Greenland. Sensitivity analysis of misclassification: a graphical and a Bayesian approach. *Annals of Epidemiology*, 16(11):834–841, 2006. doi: 10.1016/j.annepidem.2006.04.001. [p228, 229, 236]

H. Chu, H. Guo, and Y. Zhou. Bivariate random effects meta-analysis of diagnostic studies using generalized linear mixed models. *Medical Decision Making*, 30(4):499–508, 2010. doi: 10.1177/0272989X09353452. [p229, 231]

S. R. Cole, H. Chu, and S. Greenland. Multiple-imputation for measurement-error correction. *International Journal of Epidemiology*, 35(4):1074–1081, 2006. doi: 10.1093/ije/dyl097. [p228]

A. Farhi, A. D. Cohen, O. Shovman, D. Comaneshter, H. Amital, and D. Amital. Bipolar disorder associated with rheumatoid arthritis: a case-control study. *Journal of Affective Disorders*, 189:287–289, 2016. doi: 10.1016/j.jad.2015.09.058. [p229, 233, 234]

M. P. Fox, T. L. Lash, and S. Greenland. A method to automate probabilistic sensitivity analyses of misclassified binary variables. *International Journal of Epidemiology*, 34(6):1370–1376, 2005. doi: 10.1093/ije/dyi184. [p228]

S. Greenland. Basic methods for sensitivity analysis of biases. *International Journal of Epidemiology*, 25 (6):1107–1116, 1996. doi: 10.1093/ije/25.6.1107-a. [p228]

S. Greenland. Multiple-bias modelling for analysis of observational data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 168(2):267–306, 2005. doi: 10.1111/j.1467-985X.2004.00349.x. [p228]

P. Gustafson, N. D. Le, and R. Saskin. Case–control analysis with partial knowledge of exposure misclassification probabilities. *Biometrics*, 57(2):598–609, 2006. doi: 10.1111/j.0006-341X.2001.00598.x. [p228, 229]

D. Haine. **episensr***: Basic Sensitivity Analysis of Epidemiological Results*, 2021. URL https://CRAN.R-project.org/package=episensr. R package version 1.1.0. [p228]

M. D. Hoffman and A. Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014. [p233]

A. M. Jurek, G. Maldonado, S. Greenland, and T. R. Church. Exposure-measurement error is frequently ignored when interpreting epidemiologic study results. *European Journal of Epidemiology*, 21(12): 871–876, 2006. doi: 10.1007/s10654-006-9083-0. [p228]

T. L. Lash and A. K. Flink. Semi-automated sensitivity analysis to assess systematic errors in observational data. *Epidemiology*, 14(4):451–458, 2003. doi: 10.1097/01.EDE.0000071419.41011.cf. [p228]

X. Ma, L. Nie, S. R. Cole, and H. Chu. Statistical methods for multivariate meta-analysis of diagnostic tests: an overview and tutorial. *Statistical Methods in Medical Research*, 25(4):1596–1619, 2016. doi: 10.1177/0962280213492588. [p231]

R. F. MacLehose and P. Gustafson. Is probabilistic bias analysis approximately Bayesian? *Epidemiology*, 23(1):151–158, 2012. doi: 10.1097/EDE.0b013e31823b539c. [p228, 229]

I. B. McInnes and G. Schett. Pathogenetic insights from the treatment of rheumatoid arthritis. *The Lancet*, 389(10086):2328–2337, 2017. doi: 10.1016/S0140-6736(17)31472-1. [p229]

M. L. Phillips and D. J. Kupfer. Bipolar disorder diagnosis: challenges and future directions. *The Lancet*, 381(9878):1663–1671, 2013. doi: 10.1016/S0140-6736(13)60989-7. [p229]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. URL http://www.R-project.org/. [p228]

B. Rosner, W. C. Willett, and D. Spiegelman. Correction of logistic regression relative risk estimates and confidence intervals for systematic within-person measurement error. *Statistics in Medicine*, 8 (9):1051–1069, 1989. doi: 10.1002/sim.4780080905. [p228]

K. J. Rothman, S. Greenland, and T. L. Lash. *Modern Epidemiology*. Lippincott Williams & Wilkins, Philadelphia, PA, 3rd edition, 2008. [p228]

D. Spiegelman, R. J. Carroll, and V. Kipnis. Efficient regression calibration for logistic regression in main study/internal validation study designs with an imperfect reference instrument. *Statistics in Medicine*, 20(1):139–160, 2001. doi: 10.1002/1097-0258(20010115)20:1<139::AID-SIM644>3.0.CO;2-K. [p228]

Stan Development Team. **RStan***: the R Interface to Stan*, 2020. URL https://CRAN.R-project.org/package=rstan. R package version 2.21.2. [p233]

H. Wickham, W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, K. Woo, H. Yutani, D. Dunnington, and RStudio. **ggplot2***: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2021. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.5. [p233]

Jinhui Yang
University of Minnesota Twin Cities
Department of Computer Science and Engineering
200 Union St SE
Minneapolis, MN 55455
yang7004@umn.edu

Lifeng Lin
Florida State University
Department of Statistics
117 N Woodward Ave
Tallahassee, FL 32306
linl@stat.fsu.edu

Haitao Chu
University of Minnesota Twin Cities
School of Public Health
420 Delaware St SE
Minneapolis, MN 55455
chux0051@umn.edu

PAsso: an R Package for Assessing Partial Association between Ordinal Variables

by Shaobo Li, Xiaorui Zhu, Yuejie Chen, Dungang Liu

Abstract Partial association, the dependency between variables after adjusting for a set of covariates, is an important statistical notion for scientific research. However, if the variables of interest are ordered categorical data, the development of statistical methods and software for assessing their partial association is limited. Following the framework established by Liu et al. (2021), we develop an R package **PAsso** for assessing **P**artial **Asso**ciations between ordinal variables. The package provides various functions that allow users to perform a wide spectrum of assessments, including quantification, visualization, and hypothesis testing. In this paper, we discuss the implementation of **PAsso** in detail and demonstrate its utility through an analysis of the 2016 American National Election Study.

Introduction

Partial association analysis plays an important role in scientific research. It uncovers the dependency between variables after adjusting for a set of covariates, which are often considered as potential confounding factors. For continuous data, a conventional approach to assessing partial association consists of two steps: (1) regress each variable on the same set of covariates using linear regression, and (2) inspect the association between the residuals from each regression model. However, it remains challenging when the data of interest are recorded in ordinal scales. One of the challenges is that the regression models for ordinal data, such as the cumulative link models (McCullagh, 1980), do not have well-defined residuals that maintain the same properties as those from linear regression. Simply treating ordinal data as continuous and applying the conventional approach may lead to misleading results (Agresti, 2010). To this end, Liu et al. (2021) proposed to use the surrogate residuals (Liu and Zhang, 2018), a type of residual developed for ordinal regression, to assess partial associations between ordinal data. They developed a unified framework allowing quantification, hypothesis testing, and visualization of partial association. Their proposed methods can capture linear, monotonic, and non-monotonic associations. To make their methods readily and widely applicable in practice, we develop the R package **PAsso** (**P**artial **Asso**ciation). The goal of this paper is to introduce this package in both implementation and utility.

Consider a pair of ordinal variables $Y_1 = \{1, ..., J_1\}$ and $Y_2 = \{1, ..., J_2\}$, where the recorded values represent labels of ordered categories. Let $\{X_1, ..., X_p\}$ be a set of covariates to be adjusted for. We consider such a bivariate scenario in this paper unless indicated otherwise. To assess the partial association between Y_1 and Y_2, Liu et al. (2021) proposed to assess the association between their corresponding surrogate residuals (Liu and Zhang, 2018). It mimics the conventional approach as if Y_1 and Y_2 were continuous variables. Specifically, they first apply ordinal regression models, such as the cumulative link model, to Y_1 and Y_2 separately, and derive corresponding surrogate residuals R_1 and R_2. Then, assessing the partial association between Y_1 and Y_2 is equivalent to assessing the association between R_1 and R_2. The validity of this approach is supported by the key result in Liu et al. (2021), which shows that the independence between the surrogate residuals R_1 and R_2 is a sufficient and necessary condition for the partial independence between the ordinal variables Y_1 and Y_2. We provide a more detailed review of their framework subsequently in this paper.

The R package **PAsso** provides three types of association measures discussed in Liu et al. (2021). They are Pearson-correlation-based measure ϕ_ρ, Kendall-tau-based measure ϕ_τ, and a copula-based measure, Schweizer-Wolff's sigma-based ϕ_σ. Specifically,

$$\phi_\rho = \rho(R_1, R_2) = \mathrm{Cov}(R_1, R_2) / \sqrt{\mathrm{Var}(R_1)\mathrm{Var}(R_2)}; \tag{1}$$

$$\phi_\tau = \tau(R_1, R_2) = \Pr\{(R_1 - R_1^*)(R_2 - R_2^*) > 0\} - \Pr\{(R_1 - R_1^*)(R_2 - R_2^*) < 0\}; \tag{2}$$

$$\phi_\sigma = 12 \iint_{[0,1]^2} |C(u,v) - uv| du dv, \quad \textit{where } C(u,v) = \Pr\{G_1(R_1) \le u, G_2(R_2) \le v\}. \tag{3}$$

These measures are proposed in order to capture linear, monotonic, and non-monotonic relationships, respectively. For the purpose of comparison, the package also computes the Pearson correlation coefficient, Kendall's tau coefficient (Kendall, 1938), and Schweizer-Wolff's sigma (Schweizer and Wolff, 1981) for marginal associations between Y_1 and Y_2. This is convenient and useful as the marginal association is often firstly examined and set to be compared with the partial association. Furthermore, the package allows analyses for multiple variables beyond the bivariate case so that it delivers an association matrix as one of the key outputs.

In addition, bootstrap-based standard errors and p-values are reported in order to test partial independence, i.e., $H_0 : \phi = 0$, where ϕ is a general notation for the measure of partial association, including ϕ_ρ, ϕ_τ, and ϕ_σ as defined in (1)-(3). The package also provides additional flexibility to users who may wish to test whether the partial dependence is negligible at a certain threshold δ, i.e., $H_0 : |\phi| \leq \delta$ instead of $H_0 : \phi = 0$. Besides quantitative analysis outcomes, the package provides graphical tools, including partial regression plots and 3-D probability-to-probability plots (P-P plot). These graphical tools visualize the shape and strength of partial association, enabling further analysis that may help to gain extra insights. It is worth noting that the package **PAsso** also provides model diagnostic tools (Liu and Zhang, 2018), which is similar to those in the R package **sure** (Greenwell et al., 2018).

Although the focus of this paper is on ordinal data, the package **PAsso** can also deal with binary data. In fact, the binary outcome is a special case of ordered categorical data with two categories. The commonly used regression models are binary probit or logit model, for which the definition of surrogate residual remains the same. In the rest of this paper, we first review the framework established by Liu et al. (2021). Then, we provide an overview of the package **PAsso**, following which we demonstrate its utility with a real data analysis of the 2016 American National Election Study (ANES).

Review of Liu et al. (2021)'s methodology

To assess the partial association between ordinal variables, Liu et al. (2021)'s framework uses the surrogate residual (Liu and Zhang, 2018) as a key tool. Consider the cumulative link model

$$G^{-1}(P(Y \leq j)) = \alpha_j - \boldsymbol{X\beta}, \quad j = 1, 2, ..., J, \tag{4}$$

where the intercepts satisfy $-\infty = \alpha_0 < \alpha_1 < \alpha_2 < \cdots < \alpha_J = +\infty$, and $G^{-1}(\cdot)$ is a pre-specified link function. For example, $G^{-1}(u) = \Phi^{-1}(u)$ results in the ordered probit model, and $G^{-1}(u) = \log(\frac{u}{1-u})$ results in the ordered logit model, also known as proportional odds model. Model (4) can be thought of as arising from a latent variable model

$$Z = f(\boldsymbol{X}, \boldsymbol{\beta}) + \epsilon, \tag{5}$$

where $\epsilon \sim G(\cdot)$, and Z is the latent continuous variable such that $Y = j$ if $Z \in (\alpha_{j-1}, \alpha_j]$. Given the observed category of Y, Liu and Zhang (2018) defined a surrogate variable S as

$$S|(Y = j) \sim Z|(\alpha_{j-1} < Z \leq \alpha_j). \tag{6}$$

On the continuous scale of S, Liu and Zhang (2018) defined the surrogate residual R as $R = S - E(S|\boldsymbol{X})$. The statistical properties of the surrogate residual R are similar to the classical residuals defined for linear regression models. The notion also applies to other general ordinal regression models. We refer readers to Liu and Zhang (2018) for more details of the surrogate residuals.

To assess the partial association between ordinal variables Y_1 and Y_2, Liu et al. (2021) proposed to assess the association between their corresponding surrogate residuals R_1 and R_2 from ordinal regression models. This approach mimics the conventional way of assessing the partial association between continuous variables and has been justified by their key result. Specifically, conditional on a set of covariates $\boldsymbol{X} = \{X_1, ..., X_p\}$, Y_1 and Y_2 are independent if and only if their surrogate residuals R_1 and R_2 are independent. That is,

$$(Y_1 \perp\!\!\!\perp Y_2) \mid \boldsymbol{X} \Leftrightarrow R_1 \perp\!\!\!\perp R_2 \mid \boldsymbol{X}.$$

Based on this result, Liu et al. (2021) developed a new framework for assessing ordinal-ordinal partial association. The framework includes a set of new methods to quantify, visualize and test the association.

As for quantification, Liu et al. (2021) justified the advantages of using their proposed measure ϕ. First, the measure ϕ reflects the association between ordinal variables rather than that between latent continuous variables. It does not constrain itself to probit models but broadly applies to models with non-probit link functions. Its variants (1), (2), and (3) can capture linear, monotonic, and general associations. These features make the association measure ϕ fundamentally different from the polychoric correlation (Tallis, 1962), a classical association measure that describes the linear association between the latent normal variables. Moreover, their association measures do not require an upfront specification of a joint distribution of Y_1 and Y_2. Instead, their framework only requires the marginal model for each Y to be correctly specified. It thus achieved the so-called "division of labor," where the efforts of specifying marginal models and association structure are divided. This property has been seen in the development of generalized estimating equations (GEEs) and copula models. It enables us to compute the correlation matrix for a high-dimensional set of ordinal variables with affordable computational cost.

To reduce the variability caused by the simulation of the surrogate variable S in (6), Liu et al. (2021) suggested using the average

$$\hat{\phi}^{(M)} = \frac{1}{M}\sum_{m=1}^{M}\hat{\phi}^{(m)}, \tag{7}$$

where $\hat{\phi}^{(m)}$ is calculated based on the mth draw of the surrogate residuals $(R_1^{(m)}, R_2^{(m)})$ for $m = 1, ..., M$, and all draws are independent. Specifically, after fitting the two marginal regression models, we draw M independent sets of surrogate residuals for each model, i.e., $(R_1^{(1)}, ..., R_1^{(M)})$ and $(R_2^{(1)}, ..., R_2^{(M)})$. Then $\hat{\phi}^{(m)}$ is obtained based on $(R_1^{(m)}, R_2^{(m)})$, the mth draw of surrogate residuals. They numerically found that $M = 30$ is sufficient to stabilize the variance.

Liu et al. (2021) showed how to use (7) to make inferences, such as calculating standard errors, confidence intervals, and p-values. They established a bootstrap scheme to approximate the empirical distribution of $\hat{\phi}^{(M)}$ in (7). Specifically, it generates B bootstrap samples from the original data and obtains a set of bootstrap estimates $\{\hat{\phi}_1^{(M)}, \hat{\phi}_2^{(M)}, ..., \hat{\phi}_B^{(M)}\}$. Denote the empirical distribution of $\{\hat{\phi}_1^{(M)}, \hat{\phi}_2^{(M)}, ..., \hat{\phi}_B^{(M)}\}$ by $\hat{F}_B(\phi)$. This distribution approximates the distribution of $\hat{\phi}^{(M)}$. Therefore, the standard deviation of the bootstrap distribution $\hat{F}_B(\phi)$ is an estimate of standard error of $\hat{\phi}^{(M)}$. The interval $(\hat{F}_B^{-1}(\alpha/2), \hat{F}_B^{-1}(1-\alpha/2))$ can be used as a $100(1-\alpha)\%$ confidence interval. For the hypothesis testing of partial independence, i.e., $H_0 : \phi = 0$, the p-value is calculated as

$$2 \times \min(\hat{F}_B(0), 1 - \hat{F}_B(0)).$$

For example, $\hat{F}_B(0)$ is computed as $\mathbb{I}(\hat{\phi}_b^{(M)} \leq 0)/B$, where $\mathbb{I}(x)$ is the indicator function that takes value 1 if x is true and 0 otherwise. Furthermore, this approach allows testing the hypothesis whether the partial association is smaller than a threshold δ, i.e., $H_0 : |\phi| < \delta$. In this case, the p-value is calculated as $2 \times \min(\hat{F}_B(\delta), 1 - \hat{F}_B(-\delta))$. A useful application is to test the hypothesis of a negligible association, where δ usually takes a small value that can be determined by domain expert based on specific questions.

In addition to quantitative analysis, Liu et al. (2021) developed graphical tools to visualize the shape of the partial association. One of the most intuitive plots is the partial regression plot, which is a scatter plot between the surrogate residuals R_1 and R_2. Such a partial regression plot reflects the relationship between Y_1 and Y_2 after adjusting for $\boldsymbol{X}$. Another graphical tool is a 3-D probability-to-probability (P-P) plot. It is developed based on Theorem 2 and Corollary 2 in Liu et al. (2021). Specifically, if R_1 and R_2 are independent, then

$$C(u, v) = uv, \tag{8}$$

where $C(u, v)$ is a copula function for the joint distribution of the surrogate residuals R_1 and R_2, i.e., $C(u, v) = \Pr(R_1 \leq G_1^{-1}(u), R_2 \leq G_2^{-1}(v))$ where G_1 and G_2 are the assumed link functions in the model (4). The 3-D P-P plot draws $C(u, v) - uv$ against (u, v), where the deviation $C(u, v) - uv$ reflects the degree of dependence between R_1 and R_2, hence the partial dependence between Y_1 and Y_2.

In addition to the cumulative link model (4), Liu et al. (2021)'s framework applies to general ordinal regression models such as the adjacent-category logit and stereotype models. In the following sections, we introduce the **PAsso** package and discuss how to use it to implement Liu et al. (2021)'s framework with sample code and examples.

Partial association analysis with R package PAsso

An overview

Among the exported functions from the **PAsso** package, `PAsso()`, `test()`, and `plot()` are the three main functions for estimating, testing, and visualizing partial associations. In practice, users should first apply the function `PAsso()`, which generates a `PAsso` object. The components in this object include estimates of partial and marginal associations, fitted regression models for each variable, and other components to be used for relevant analysis. The generated `PAsso` object plays an instrumental role, as it is a necessary input for other functions in the package. This design provides convenience as the details of data, regression models, and type of association measures need only to be specified once in the function PAsso(), and they will be passed to other functions once the PAsso object is called.

In addition to the main functions, the **PAsso** package also provides several other functions that can further assist the analysis of partial association. Specifically, the function plot3D() produces the copula-based 3-D P-P plot, which helps to visualize the strength of a general partial association. The function residuals() can extract the surrogate residuals from a PAsso object or a model object. The function diagnostic.plot() generates residual-based model diagnostic plots to help validate the model specifications.

We provide in Figure 1 a flow chart to illustrate how to use the **PAsso** package for assessing partial association under a bivariate setting. On the far left, the grey block contains elements to input: the data and the models. By calling the function PAsso(), we generate a PAsso object as shown in the orange block in the center. This PAsso object can yield results for estimation, testing, and visualization (the three orange blocks on the far right). The PAsso object can also give us residuals, which can be used for model diagnostics (blue blocks). In fact, the model diagnostic plots are byproducts of the package. To make it more convenient, we provide the function diagnostic.plot() that can be directly applied to the PAsso object.

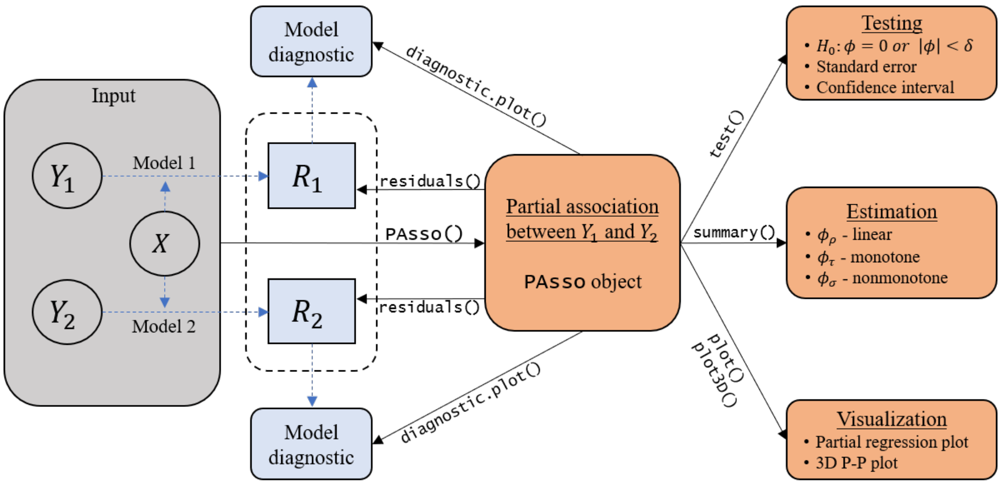


Figure 1: Illustration of functions in the **PAsso** package for partial association analysis.

Figure 1 clearly shows that the PAsso object plays a central role in the entire analysis, where all the other functions take it as an input. Therefore, understanding the PAsso() function and specifying appropriate inputs are important, which ensures valid results produced by other functions in the subsequent analysis. Next, we describe the detailed implementation and the key inputs of the function PAsso(), as well as other functions in the package.

Implementation of main functions

PAsso() for estimation

To apply the function PAsso(), users need to specify the data frame, the names of Y variables (e.g., Y_1 and Y_2) and covariates, and the type of association measure. Other inputs, including the type of model and residual method, have default values. The first step performed by PAsso() is to estimate the regression model for each Y variable. Based on the data type, by default, the function PAsso() applies the cumulative probit model as in (4) (polr() from **MASS** (Venables and Ripley, 2002)) if Y is ordinal and the binary probit model (glm() from **stats**) if Y is binary. Users can also specify other link functions supported by polr() and glm().

Once the regression models for Y_1 and Y_2 are estimated, the surrogate residuals, R_1 and R_2, can be obtained by using the function residuals() in the package. Here, the function residuals() calls the fitted model based on which the surrogate residuals are computed. The implementation of residuals() is similar to the resids() function from package **sure**. Then, by applying the specified measure to the surrogate residuals, one can obtain the estimated partial association defined in (7). In the current package version, the three types of association measures, (1), (2), and (3), are implemented with cor() from the **stats** package, cor.fk() from the **pcaPP** package (Filzmoser et al., 2018), and wolfCOP() from the **copBasic** package (Asquith, 2020), respectively.

It is worth mentioning that the function PAsso() provides additional flexibility by allowing pre-fitted model objects to be the inputs. It permits users to use specific models based on domain knowledge and experiences. Similarly, the pre-fitted model objects can also be the inputs for the functions `residuals()` and `diagnostic.plot()`. The supported models and corresponding R functions along with the packages (**rms** (Harrell Jr, 2019), **ordinal** (Christensen, 2019), **VGAM** (Yee et al., 2010)) are listed in Table 1.

		stats	MASS	rms		ordinal	VGAM	
Response	Model	`glm`	`polr`	`lrm`	`orm`	`clm`	`vglm`	`vgam`
Ordinal	Ordered probit		✓		✓	✓	✓	✓
	Ordered logit (proportional odds)		✓		✓	✓	✓	✓
	Proportional hazard (clog-log)		✓		✓	✓	✓	✓
	Adjacent-category logit						✓	✓
Binary	Logit	✓		✓			✓	✓
	Probit	✓					✓	✓
	Complementary log-log	✓					✓	✓

Table 1: Models and packages supported by **PAsso**.

`test()` for hypothesis testing

To carry out more detailed inferences on the estimated partial association, one can apply the function `test()`, which is implemented based on a bootstrap scheme discussed earlier. To be specific, it first generates bootstrap samples based on the original data and then repeats the calculations performed in the `PAsso()` function for each bootstrap sample. As a result, the bootstrap distribution of $\hat{\phi}^{(M)}$ defined in (7) can be obtained.

The components from `test()` include standard errors, confidence intervals, and p-values for the partial independence test. To apply `test()`, in addition to the key input `PAsso` object, the user can specify the number of bootstrap samples, the type of null hypothesis, and whether or not parallel computing should be employed. If the Schweizer-Wolff-sigma-based measure ϕ_σ is used, the computational cost of `test()` may be high. This is because the estimation of ϕ_σ based on the current version of the package **copBasic** is slow due to the double integration in the formula (3).

`plot()` and `plot3D()` for visualization

There are two types of plots being covered in the **PAsso** package: the pairwise partial regression plot and the 3-D P-P plot. The pairwise partial regression plot can be obtained by applying the function `plot()`. It is implemented based on the function `ggpairs()` in the package **GGally** (Schloerke et al., 2020), which is a nice extension of the widely used graphical tool **ggplot2** (Wickham, 2016). For users who are familiar with **GGally**, the produced figure can be fully customized using the arguments for the function `ggpairs()`.

The 3-D P-P plot is obtained by using the function `plot3D()`. This function is implemented based on the **plotly** (Sievert, 2020) package, which offers interactive graphs. It plots $C(u,v) - uv$ against u and v as discussed in (8). The height of the surface reflects the degree of partial dependence between Y_1 and Y_2.

Table 2 summarizes the main input and output of the functions in the **PAsso** package. Users should refer to the package vignette for detailed arguments and output values.

Analysis of the 2016 American National Election Study (ANES)

We demonstrate the utility of the **PAsso** package with a real data example, a sample of the survey data from the 2016 American National Election Studies (the data of Time Series study). The American National Election Studies (ANES) is a joint project between the University of Michigan and Stanford University since 1948 (DeBell, 2010), aiming to provide researchers, policymakers, and citizens with high-quality survey data pertinent to political science. The original data include over a thousand survey questions for pre- and post-election surveys based on the same population (4,271 respondents). For our analysis, we select eight survey questions in the pre-election study and remove the respondents who have missing values on any of these eight variables. Here, we assume that the missing values are missing completely at random. As a result, our sample consists of 2,188 respondents. Table 3 describes the variables in our sample.

To demonstrate the utility of the **PAsso** package, we conduct an illustrative analysis, which focuses on the four variables: voter's party identification (PID), his/her own left-right placement (selfLR)

Function name	Functionality	Input	Output
`PAsso()`	Estimation	• dataframe to be analyzed; • names of the Y variables; • names of the covariates; • association measure; • other inputs such as marginal model specifications have default values (see the package vignette).	• partial and marginal association matrix; • summary of model coefficient estimates for each marginal model; • results also include surrogate residuals and detailed model summaries.
`test()`	Testing	• the `PAsso` object; • number of bootstrap samples; • type of null hypothesis.	• Partial association matrix; • standard errors; • p-values; • confidence intervals.
`plot()`	Visualization	• the `PAsso` object; • other arguments follow the **GGally** (see the package vignette).	• Pairwise partial regression plot.
`plot3D()`	Visualization	• the `PAsso` object; • two variable names; • other arguments follow the **plotly** (see the package vignette).	• 3-D P-P plot based on Copula.
`residuals()`	Residuals	• the `PAsso` object or a single model object; • number of draws; • residual method.	• surrogate residuals.
`diagnostic.plot()`	Diagnostics	• the `PAsso` object or a single model object.	• model diagnostic plots.

Table 2: Summary of inputs and outputs of the functions in **PAsso**.

Variable	Explanation
age	respondent's age
education	respondent's highest level of education. We create another variable 'edu.year', which recodes the original education to a continuous scale as the approximate number of years of education. (This conversion is for convenience in model fitting.)
income	respondent's annual income in categories, e.g., \$40k-\$60k. We create another variable 'income.num', which recodes original income to continuous scale (the median of the recorded range).
PID	respondent's party identification with 7 ordinal levels from strong democrat (=1) to strong republication (=7).
selfLR	respondent's self-placement about own left-right placement in 7 ordinal levels from extremely liberal (=1) to extremely conservative (=7).
TrumpLR	respondent's opinion about Donald Trump's left-right placement in 7 ordinal levels (same scale as selfLR).
ClinLR	respondent's opinion about Hilary Clinton's left-right placement in 7 ordinal levels (same scale as selfLR).
PreVote	respondent's voting preference between Donald Trump and Hilary Clinton.

Table 3: Variable description for the selected sample of the 2016 ANES data.

and the left-right placement for Donald Trump (TrumpLR) and Hilary Clinton (ClinLR). These four variables of interest have the same ordinal scale, and they all represent respondents' political views for themselves as well as for the two presidential candidates. We attempt to answer the following question. Are these four variables still correlated after adjusting for their age, education, and income?

In what follows, we use the **PAsso** package to answer this question step-by-step from three aspects: estimation, hypothesis testing, and graphical visualization.

Estimation

First, we use the function `PAsso()` to obtain the estimates of the association measures in (1)-(3). We specify the names of these four variables as `responses` and the covariate names as the `adjustments`. If the Kendall-tau-based measure (2) is desired, we can specify the option `method="kendall"`, as in the sample code below.

```
library(PAsso)
data(ANES2016)   # load the dataset
set.seed(2020)  # for reproducibility
phi <- PAsso(responses = c("PID", "selfLR", "TrumpLR", "ClinLR"),
             adjustments = c("age", "edu.year", "income.num"),
             data = ANES2016,
             method = "kendall")
print(phi, digits = 3)

#> --------------------------------------------
#> The partial correlation coefficient matrix:
#>         PID     selfLR  TrumpLR  ClinLR
#> PID      1.000   0.516  -0.061   -0.323
#> selfLR           1.000  -0.103   -0.294
#> TrumpLR                  1.000   -0.072
#> ClinLR                            1.000
```

As shown above, the default output is pairwise correlations in a matrix form. The `PAsso` object contains other hidden components, including the marginal association matrix, details of each regression model, and all draws of the surrogate residuals. For users' convenience, we also make the generic function `summary()` available for the `PAsso` object.

```
summary(phi, digits = 4)

#> --------------------------------------------
#> The partial correlation coefficient matrix:
#>
#>         PID      selfLR   TrumpLR  ClinLR
#> PID      1.0000   0.5161  -0.0610  -0.3232
#> selfLR            1.0000  -0.1034  -0.2938
#> TrumpLR                    1.0000  -0.0715
#> ClinLR                              1.0000
#> --------------------------------------------
#> The marginal correlation coefficient matrix:
#>
#>         PID      selfLR   TrumpLR  ClinLR
#> PID      1.0000   0.6331  -0.0956  -0.4081
#> selfLR            1.0000  -0.1592  -0.3724
#> TrumpLR                    1.0000  -0.0721
#> ClinLR                              1.0000
#>
#> --------------------------------------------
#> --------------------------------------------
#>
#> The coefficients of fitted models are:
#>
#>             PID         selfLR      TrumpLR     ClinLR
#> age          0.0048***   0.0098***  -0.0056***  -0.0069***
#> Std. Error   0.0013      0.0013      0.0013      0.0013
#> ---
#> edu.year    -0.0459***  -0.0737***   0.0540***  -0.0127
#> Std. Error   0.0098      0.0095      0.0096      0.0097
#> ---
#> income.num   0.0009*     0.0004      0.0005     -0.0009*
#> Std. Error   0.0004      0.0004      0.0004      0.0004
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The function `summary()` gives the partial and marginal association in two separate matrices and summary of coefficient estimates of each regression model. The comparison between the two association matrices shows the extent to which the strength of association has been weakened due to the adjusted covariates. Clearly, the associations between all pairs of the four variables have been weakened by as much as 36% (PID vs. TrumpLR) and as less as 0.8% (TrumpLR vs. ClinLR). The last table is the model summary table, where each column displays the coefficient estimates and their standard errors for each regression model, and the column names are the names of response variables (e.g., PID, selfLR, TrumpLR, and ClinLR). The stars represent the range of p-values as noted at the bottom of the model summary table. For instance, `'***'` indicates that the corresponding p-value is between 0 and 0.001.

Hypothesis testing

Next, we use the function `test()` to obtain the p-values and bootstrap standard errors of the partial associations. This function is directly applied to the `PAsso` object. The number of bootstrap replicates can be specified with the argument `bootstrap_rep`, whose default value is 300. In general, we recommend the minimum number of bootstrap replicates to be 1000 in order to carry out more accurate p-values. To speed up the computational time, we can further adopt parallel computing by specifying `parallel=TRUE`. However, this option requires users to pre-set the number of cores and the cluster, as demonstrated below.

```
library(doParallel) # load packages for parallel computing
library(progress)
numCores <- detectCores()  # Number of CPU cores. Do Not be too aggressive!
# Set up parallel backend (multicore for unix and snow for windows)
cl <- if (.Platform$OS.type == "unix") numCores else makeCluster(numCores)
registerDoParallel(cl)
test(phi, bootstrap_rep = 1000, H0 = 0, parallel = TRUE)

# The partial association analysis:
#
#          PID      selfLR    TrumpLR   ClinLR
# PID       1.0000   0.5161   -0.0610   -0.3232
# S.E.              0.0094    0.0134    0.0102
# Pr                0.001***  0.001***  0.001***
# ---
# selfLR             1.0000   -0.1034   -0.2938
# S.E.                        0.0129    0.0108
# Pr                          0.001***  0.001***
# ---
# TrumpLR                      1.0000   -0.0715
# S.E.                                  0.0154
# Pr                                    0.001***
# ---
# ClinLR                                1.0000
# S.E.
# Pr
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The obtained p-values indicate that the partial associations between each pair of variables are statistically significant. We can also test for negligible dependence with a certain threshold δ. In this case, the null hypothesis is $H_0 : |\phi| \leq \delta$. The value of δ can be specified using the argument `H0`. The default value of `H0` is 0, representing $H_0 : \phi = 0$. The example below illustrates a negligible dependence test with threshold $\delta = 0.05$.

```
test(phi, bootstrap_rep = 1000, H0 = 0.05, parallel = TRUE)

# The partial association analysis:
#
#          PID      selfLR    TrumpLR   ClinLR
# PID       1.0000   0.5161   -0.0610   -0.3232
# S.E.              0.0093    0.0135    0.0104
# Pr                0.001***  0.450     0.001***
# ---
# selfLR             1.0000   -0.1034   -0.2938
# S.E.                        0.0133    0.0109
# Pr                          0.001***  0.001***
# ---
# TrumpLR                      1.0000   -0.0715
# S.E.                                  0.0153
# Pr                                    0.128
# ---
# ClinLR                                1.0000
# S.E.
# Pr
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the output, we can see that the partial associations between PID and TrumpLR, and TrumpLR and ClinLR are not significant. This result implies that the partial dependencies can be negligible under the threshold $\delta = 0.05$.

Visualization

To further understand how these variables are associated after the adjustment, we utilize graphical tools for quick visualization. First, we apply the function `plot()` to the `PAsso` object `phi`. It produces the pairwise partial regression plot.

```
plot(phi, color = "red", alpha = 0.1) # alpha specifies opacity
```

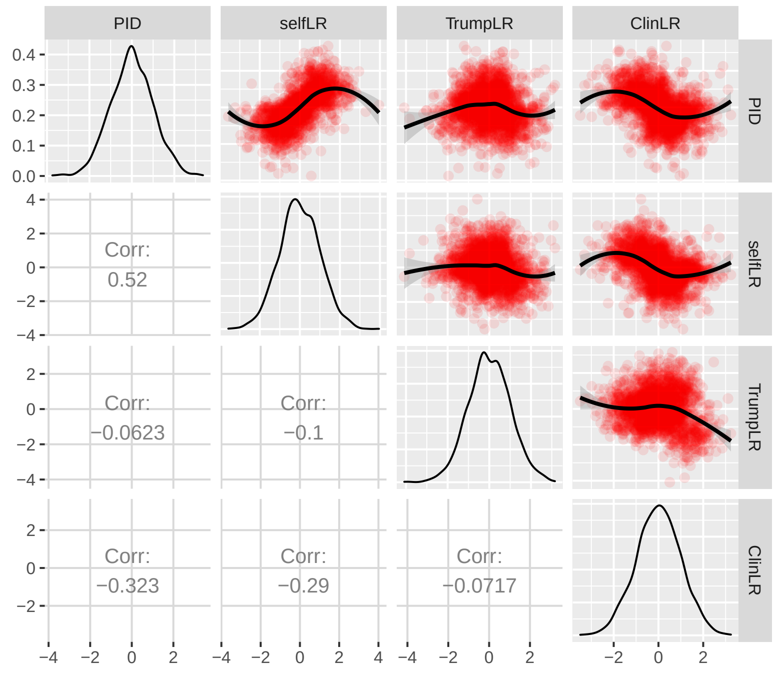


Figure 2: Pairwise partial regression plot for PID, selfLR, TrumpLR, and ClinLR.

Figure 2 contains the partial regression plots displayed on the upper triangle, the estimated partial associations shown on the lower triangle, and the density plots of the surrogate residuals on the diagonal line. The partial regression plot between each pair of variables is essentially the scatter plot of the corresponding pair of surrogate residuals. The fitted smooth curve on top of each scatter plot is through the local weighted smoothing splines (LOWESS) that are available in the GGally package.

An interesting finding from Figure 2 is that the weakest partial association (between PID and TrumpLR) shown previously may not necessarily be weak because a quadratic relationship is clearly shown. To confirm this relationship, it is worth drawing the 3-D P-P plot, which can indicate the strength of a non-monotonic relationship. This can be done by applying the function `plot3D()` on the `PAsso` object and specifying the pair of variables of interest.

```
plot3D(phi, y1 = "PID", y2 = "TrumpLR")
```

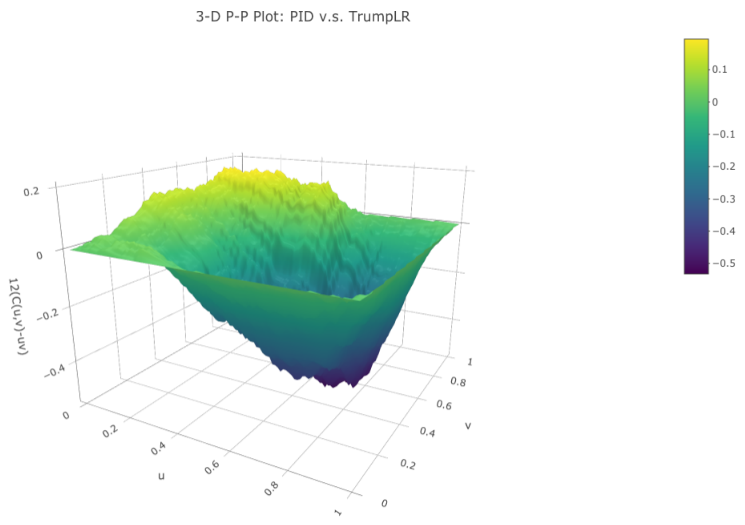


Figure 3: 3D probability-to-probability plot based on a bivariate copula.

From Figure 3, the surface reaches both positive and negative sides (vertical axis). This is consistent with the quadratic pattern displayed in the partial regression plot in Figure 2. Furthermore, the altitude of the surface on both sides is moderate, indicating that the strength of the partial association between PID and TrumpLR may not be negligible. Additionally, the function `plot3D()` also provides an option to convert the 3-D plot to a contour plot by specifying `type = "contour"`. The following line of code creates Figure 4, showing the contour plot corresponding to Figure 3.

```
plot3D(phi, y1 = "PID", y2 = "TrumpLR", type = "contour")
```

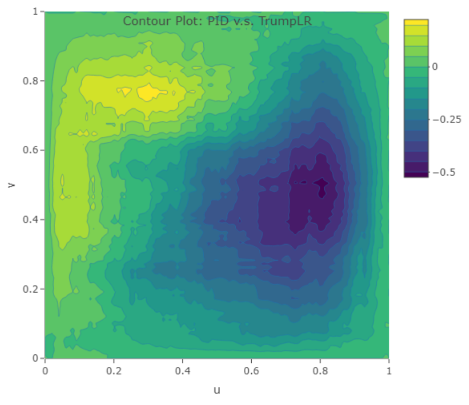


Figure 4: Colored contour plot based on a bivariate copula.

As mentioned earlier, the function `PAsso()` allows users to use pre-fitted regression models as inputs. Below, we provide a sample code for this alternative use of `PAsso()`.

```
library(MASS)
# convert numeric response to factor
Yname <- c("PID", "selfLR", "TrumpLR", "ClinLR")
ANES2016[Yname] <- lapply(ANES2016[Yname], factor)
# fit each model separately
fit.PID <- polr(PID ~ age + edu.year + income.num, data = ANES2016, method = "probit")
fit.selfLR <- polr(selfLR ~ age + edu.year + income.num, data = ANES2016, method = "probit")
fit.TrumpLR <- polr(TrumpLR ~ age + edu.year + income.num, data = ANES2016, method = "probit")
fit.ClinLR <- polr(ClinLR ~ age + edu.year + income.num, data = ANES2016, method = "probit")
# assess partial association
set.seed(2020)
phi1 <- PAsso(fitted.models = list(fit.PID, fit.selfLR, fit.TrumpLR, fit.ClinLR),
              method = "kendall")
print(phi1, digits = 3)
```

```
#> -------------------------------------------
#> The partial correlation coefficient matrix:
#>          PID     selfLR  TrumpLR  ClinLR
#> PID       1.000   0.516  -0.061   -0.323
#> selfLR            1.000  -0.103   -0.294
#> TrumpLR                   1.000   -0.072
#> ClinLR                             1.000
```

The above output is exactly the same as the other way we showed earlier. Such flexibility allows user to specify different models and link functions for different response variables based on domain expertise and convention, while it does not affect other functions used in the following analyses. Each of these model objects (`fit.PID`, `fit.selfLR`, `fit.TrumpLR`, and `fit.ClinLR`) can also be the input for the functions `residuals()` and `diagnostic.plot()`.

Model diagnostics

In addition to assessing partial association, the package also provides the function `diagnostic.plot()` for model diagnostics. The output of this function contains three types of diagnostic plots: (1) residual Q-Q plot; (2) residual vs. fitted value; and (3) residual vs. covariates. We can specify one of these types of the plot by using the argument `output`. Since there are at least two regression models being estimated, the function `diagnostic.plot()` allows the user to specify a particular model (using the argument `model_id`) for which the diagnostic plots are produced. We provide two examples below to illustrate the two different outputs.

```
# residual vs. fitted value (the linear predictor) for all models
diagnostic.plot(phi, output = "fitted")
```

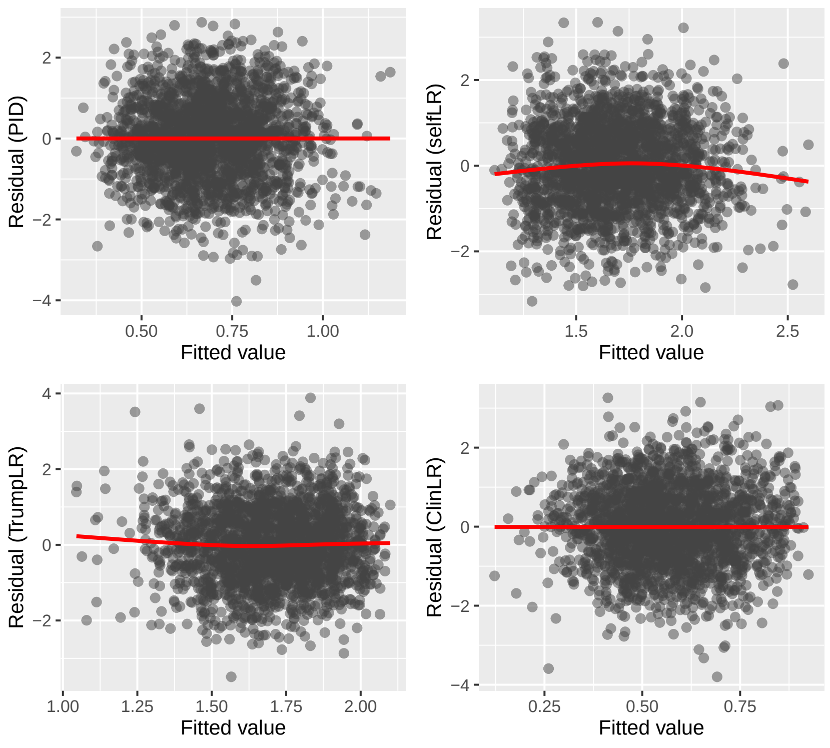


Figure 5: Model diagnostic plot (Residual vs. fitted) for all four models.

```
# residual Q-Q plot for the second model (selfLR)
diagnostic.plot(phi, output = "qq", model_id = 2)
```

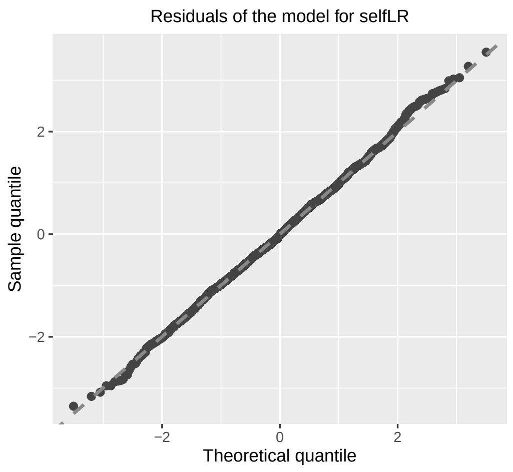


Figure 6: Model diagnostic plot (Residual Q-Q plot) for the second model.

Figure 5 shows the scatter plot between surrogate residuals and the fitted values (the linear predictor in the model (5)) for all four models. Figure 6 shows the Q-Q plot for the second model whose response variable is "selfLR." Neither of the two figures provides evidence of model misspecification or inadequate fitting.

Adjacent-category logit model

Finally, we show an example where the adjacent-category logit model is employed for the covariates adjustment. In addition, we also include a binary variable, "PreVote", in order to demonstrate the utility of **PAsso** for different types of data. The variable "PreVote" takes two values: Hilary Clinton and Donald Trump. For convenience, we recode "PreVote" to numeric values 0 (Hilary Clinton) and 1 (Donald Trump), and the binary logit model is used. For the demonstration purpose, we only show the estimation results from `PAsso()`.

```
ANES2016$PID <- as.numeric(ANES2016$PID)
phi2 <- PAsso(responses = c("PreVote.num", "PID", "selfLR", "TrumpLR", "ClinLR"),
              adjustments = c("age", "edu.year", "income.num"),
              data = ANES2016,
              method = "kendall",
              model = c("logit", "acat", "acat", "acat", "acat"))
summary(phi2, digits = 3)

#> -------------------------------------------
#> The partial correlation coefficient matrix:
#>
#>              PreVote.num  PID     selfLR  TrumpLR  ClinLR
#> PreVote.num   1.000        0.445   0.390  -0.091   -0.300
#> PID                        1.000   0.516  -0.062   -0.319
#> selfLR                             1.000  -0.105   -0.293
#> TrumpLR                                    1.000   -0.071
#> ClinLR                                              1.000
#> -------------------------------------------
#> The marginal correlation coefficient matrix:
#>
#>              PreVote.num  PID     selfLR  TrumpLR  ClinLR
#> PreVote.num   1.000        0.706   0.637  -0.182   -0.494
#> PID                        1.000   0.633  -0.096   -0.408
#> selfLR                             1.000  -0.159   -0.372
#> TrumpLR                                    1.000   -0.072
#> ClinLR                                              1.000
#>
```

```
#> --------------------------------------------
#> --------------------------------------------
#>
#> The coefficients of fitted models are:
#>
#>              PreVote.num  PID        selfLR     TrumpLR    ClinLR
#> age           0.015***    -0.002***  -0.006***   0.002**   0.004***
#> Std. Error    0.003        0.001      0.001      0.001     0.001
#> ---
#> edu.year     -0.129***     0.020***   0.045***  -0.036***  0.022**
#> Std. Error    0.019        0.004      0.006      0.006     0.007
#> ---
#> income.num    0.001        0.000**    0.000     -0.001**   0.001**
#> Std. Error    0.001        0.000      0.000      0.000     0.000
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The same elements as for the cumulative link model are printed: the estimated partial and marginal associations and the coefficient estimates of the five models. As specified in the argument `model`, the binary logit model is used for `PreVote`, and the adjacent-category logit model ("acat") is used for the other four variables. From the output, we found that the strength of association between "PreVote" and "PID" reduces from 0.706 to 0.444 ($\approx$ 37%) after adjusting for the three covariates. This result is qualitatively consistent with that in Liu et al. (2021), which has conducted the same analysis based on the 1996 ANES data.

Summary

We have developed the R package **PAsso** (Zhu et al., 2021) for assessing the partial association between ordinal variables. The development follows the statistical framework established by Liu et al. (2021). The package provides several functions, allowing estimation, hypothesis testing, and visualization of partial associations. By using this single package, users can quickly obtain a full picture of the partial associations between multiple variables.

Bibliography

A. Agresti. *Analysis of Ordinal Categorical Data*. John Wiley & Sons: Hoboken, New Jersey, 2 edition, 2010. [p239]

W. Asquith. *copBasic—General Bivariate Copula Theory and Many Utility Functions*, 2020. R package version 2.1.5. [p242]

R. H. B. Christensen. ordinal—regression models for ordinal data, 2019. R package version 2019.12-10. https://CRAN.R-project.org/package=ordinal. [p243]

M. DeBell. How to analyze anes survey data. *Am. Nat'l Election Stud.(May 2010), http://electionstudies.org/resources/papers/neso12492. pdf [http://perma. ee/Y4VG-X47F]*, 2010. [p243]

P. Filzmoser, H. Fritz, and K. Kalcher. *pcaPP: Robust PCA by Projection Pursuit*, 2018. URL `https://CRAN.R-project.org/package=pcaPP`. R package version 1.9-73. [p242]

B. M. Greenwell, A. J. McCarthy, B. C. Boehmke, and D. Liu. Residuals and diagnostics for binary and ordinal regression models: An introduction to the sure package. *The R Journal*, 10(1):381–394, 2018. [p240]

F. E. Harrell Jr. *rms: Regression Modeling Strategies*, 2019. URL `https://CRAN.R-project.org/package=rms`. R package version 5.1-4. [p243]

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. [p239]

D. Liu and H. Zhang. Residuals and diagnostics for ordinal regression models: A surrogate approach. *Journal of the American Statistical Association*, 113(522):845–854, 2018. [p239, 240]

D. Liu, S. Li, Y. Yu, and I. Moustaki. Assessing partial association between ordinal variables: quantification, visualization, and hypothesis testing. *Journal of the American Statistical Association*, 116(534): 955–968, 2021. doi: 10.1080/01621459.2020.1796394. [p239, 240, 241, 251]

P. McCullagh. Regression models for ordinal data (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 42:109–142, 1980. [p239]

B. Schloerke, J. Crowley, D. Cook, F. Briatte, M. Marbach, E. Thoen, A. Elberg, and J. Larmarange. *GGally: Extension to 'ggplot2'*, 2020. URL `https://CRAN.R-project.org/package=GGally`. R package version 1.5.0. [p243]

B. Schweizer and E. F. Wolff. On nonparametric measures of dependence for random variables. *The Annals of Statistics*, 9(4):879–885, 1981. [p239]

C. Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN 9781138331457. URL `https://plotly-r.com`. [p243]

G. Tallis. The maximum likelihood estimation of correlation from contingency tables. *Biometrics*, 18(3): 342–353, 1962. [p240]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `https://www.stats.ox.ac.uk/pub/MASS4/`. ISBN 0-387-95457-0. [p242]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2016. [p243]

T. W. Yee et al. The vgam package for categorical data analysis. *Journal of Statistical Software*, 32(10): 1–34, 2010. [p243]

X. Zhu, S. Li, Y. Chen, and D. Liu. *PAsso: Assessing the Partial Association Between Ordinal Variables*, 2021. URL `https://CRAN.R-project.org/package=PAsso`. R package version 0.1.10. [p251]

Shaobo Li
University of Kansas
1654 Naismith Drive
Lawrence, KS 66045
`shaobo.li@ku.edu`

Xiaorui Zhu
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
`zhuxr@mail.uc.edu`

Yuejie Chen
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
`chen4y9@mail.uc.edu`

Dungang Liu
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
`dungang.liu@uc.edu`

bcmixed: A Package for Median Inference on Longitudinal Data with the Box–Cox Transformation

by Kazushi Maruo, Ryota Ishii, Yusuke Yamaguchi, Masahiko Gosho

Abstract This article illustrates the use of the **bcmixed** package and focuses on the two main functions: `bcmarg` and `bcmmrm`. The `bcmarg` function provides inference results for a marginal model of a mixed effect model using the Box–Cox transformation. The `bcmmrm` function provides model median inferences based on the mixed effect models for repeated measures analysis using the Box–Cox transformation for longitudinal randomized clinical trials. Using the `bcmmrm` function, analysis results with high power and high interpretability for treatment effects can be obtained for longitudinal randomized clinical trials with skewed outcomes. Further, the **bcmixed** package provides summarizing and visualization tools, which would be helpful to write clinical trial reports.

Introduction

Longitudinal data are often observed in medical or biological research. One of the most popular statistical models for studying longitudinal continuous outcomes is the linear mixed effect model. Several packages are available from CRAN that allow for the implementation of linear mixed effects models (e.g., **nlme** (Pinheiro et al., 2021), **glme** (Sam Weerahandi et al., 2021), **lme4** (Bates et al., 2015), **CLME** (Jelsema and Peddada, 2016), **PLmixed** (Rockwood and Jeon, 2019), **MCMCglmm** (Hadfield, 2010)).The linear mixed effect models assume that longitudinal outcomes follow a multivariate normal distribution. However, the distribution of the outcome is often right skewed in the medical and biological fields. Therefore, evaluating fixed effects based on the normal distribution theory may result in inefficient inferences such as power loss for some statistical tests. In addition, although a model-based mean for a certain level of the categorical exploratory variables is often estimated when applying the linear mixed effect model (e.g., the model-based mean for each treatment group of the last visit in a randomized clinical trial), the mean may be inadequate as a representative value for the skewed data.

The Box–Cox transformation (Box and Cox, 1964) is often applied to skewed longitudinal data (Lipsitz et al., 2000) to reduce the skewness of a skewed outcome and apply existing statistical models based on a normal distribution. However, it is difficult to directly interpret the model mean estimated on the scale after applying some transformations to the outcome variable.

For the sake of the interpretability of the analysis results, Maruo et al. (2015) propose a model median inference method on the original scale based on the Box–Cox transformation in the context of randomized clinical trials. Maruo et al. (2017) extend this method to the framework of mixed effects models for repeated measures (MMRM) analysis (Mallinckrodt et al., 2001) in the context of longitudinal randomized clinical trials.

The **bcmixed** package (Maruo et al., 2020) contains functions to estimate model medians for longitudinal data proposed by Maruo et al. (2017), as well as a sample data set that is used in Maruo et al. (2017). In this package, the parameter estimation is conducted partially using the `gls` function in the **nlme** package. This paper illustrates the usage of the package with the sample data in several contexts.

The remainder of this manuscript is organized as follows: In section Methods, we describe the methods proposed by Maruo et al. (2017). Then, we illustrate the usage of the **bcmixed** package with the example data in section Illustrations. Finally, our contributions are summarized in section Summary and discussion.

Methods

We briefly introduce the method proposed by in Maruo et al. (2017). The detailed expressions can be found in Maruo et al. (2017).

Parameter inference for the Box–Cox model

Let the outcome be a continuous and positive value. The outcomes are measured over time for each subject $i = 1, \dots, n$, and the number of planned measurement occasions is T (occasion index: $t = 1, \dots, T$). The outcome vector for the ith subject is denoted by $\boldsymbol{y}_i = (y_{i1}, \dots, y_{in_i})^T$, where n_i is the number of measurements for the ith subject. We have $n_i \leq T$ because of missingness. Then, we consider applying the following model (Lipsitz et al., 2000; Box and Cox, 1964):

$$\boldsymbol{z}_i = X_i\boldsymbol{\beta} + W_i\boldsymbol{b}_i + \boldsymbol{\epsilon}_i, \tag{1}$$

where $\boldsymbol{z}_i$ is the Box–Cox transformed outcome vector such that the jth element ($j = 1, \dots, n_i$) is denoted by

$$z_{ij} = \begin{cases} (y_{ij}^{\lambda} - 1)/\lambda, & \lambda \neq 0, \\ \log y_{ij}, & \lambda = 0, \end{cases}$$

and λ is the transformation parameter. The parameter $\boldsymbol{\beta}$ is the p-dimensional vector of the fixed effect, $\boldsymbol{b}_i$ is the q-dimensional vector of random effects distributed as $MVN_q(\boldsymbol{0}_q, D)$, $\boldsymbol{\epsilon}_i$ is the n_i-dimensional vector of random errors distributed independently as $MVN_{n_i}(\boldsymbol{0}_{n_i}, \Sigma_i)$, and X_i and W_i are $n_i \times p$ and $n_i \times q$ design matrices relating the fixed and random effects, respectively. The random variables $\boldsymbol{b}_i$ and $\boldsymbol{\epsilon}_i$ are independent. From Formula (1), it is derived that $\boldsymbol{z}_i|\lambda \sim MVN_{n_i}(X_i\boldsymbol{\beta}, V_i)$, where $V_i = W_iDW_i^T + \Sigma_i$. In this paper, we consider situations where researchers have little interest in random effects and are interested in assessing fixed effects. In such cases, a simple formulation of the linear mixed effect model (1) can be implemented wherein the random effects are not explicitly modeled, but are rather included as part of the covariance matrix V_i. We focus on such a "marginal" mean model. The covariance parameter vector in $V = V_i$ for $n_i = T$ (i.e., subjects with no missing values) is denoted as $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)^T$. The dimension of $\boldsymbol{\alpha}$, that is m, depends on T and the specified covariance structure.

Let the model parameter vector be $\boldsymbol{\theta} = (\lambda, \boldsymbol{\beta}^T, \boldsymbol{\alpha}^T)^T$. The maximum likelihood estimate of $\boldsymbol{\theta}$ is obtained through maximizing the profile likelihood for λ (Lipsitz et al., 2000; Maruo et al., 2017). The model-based and robust variance estimators of the maximum likelihood estimator $\hat{\boldsymbol{\theta}}$ are given by $V_{\boldsymbol{\theta}}^{(M)} = \{-\hat{H}\}^{-1}$ and $V_{\boldsymbol{\theta}}^{(R)} = \{-\hat{H}\}^{-1}\hat{J}\{-\hat{H}\}^{-1}$, respectively, where $H = \frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T} l(\boldsymbol{\theta})$, $J = \sum_{i=1}^{n} \left\{\frac{\partial}{\partial\boldsymbol{\theta}} l_i(\boldsymbol{\theta})\right\} \left\{\frac{\partial}{\partial\boldsymbol{\theta}} l_i(\boldsymbol{\theta})\right\}^T$, $l(\boldsymbol{\theta})$ is the likelihood function for n subjects, and $l_i(\boldsymbol{\theta})$ is the likelihood function for the ith subject. The matrices $\hat{H}$ and $\hat{J}$ are obtained from the matrices H and J by replacing $\boldsymbol{\theta}$ by $\hat{\boldsymbol{\theta}}$, respectively. A robust variance estimator is an asymptotically valid estimator even when the model (1) is mis-specified (Cox, 1961).

Model median inference

We now focus on the continuous and positive outcomes of a certain disease, and consider a situation in which the efficacy of some treatments (group index: $g = 1, ..., G$) is compared based on a randomized, parallel group clinical trial, where the total number of subjects is n. The explanatory variable matrix, X_i, and the fixed effect parameter, $\boldsymbol{\beta}$ in model (1) are denoted in detail as follows. Now, X_i is given by the $n_i \times (GT + C)$ matrix that contains the intercept, $G - 1$ group variables, $T - 1$ occasion variables, group-by-occasion interaction variables, and C covariates, where the categorical covariates are converted into dummy variables. The fixed effect parameter vector is given by $\boldsymbol{\beta} = (\beta_I, \boldsymbol{\beta}_g^T, \boldsymbol{\beta}_t^T, \boldsymbol{\beta}_{gt}^T, \boldsymbol{\beta}_c^T)^T$, where β_I, $\boldsymbol{\beta}_g = (\beta_{g(1)}, \dots, \beta_{g(G-1)})^T$, $\boldsymbol{\beta}_t = (\beta_{t(1)}, \dots, \beta_{t(T-1)})^T$, $\boldsymbol{\beta}_{gt} = (\beta_{gt(1,1)}, \beta_{gt(1,2)}, \dots, \beta_{gt(G-1,T-1)})^T$, and $\boldsymbol{\beta}_c = (\beta_{c(1)}, \dots, \beta_{c(C)})^T$ are the intercept, group, occasion, group-by-occasion, and covariate parameter vectors, respectively. Although group G and occasion T is set to be at the reference level, we set $\beta_{g(G)} = \beta_{t(T)} = \beta_{gt(G,t)} = \beta_{gt(g,T)} = 0$ for the sake of description ($g = 1, \dots, G$, $t = 1, \dots, T$).

Under these settings, the model median, $\xi_{(g,t)}$, for the treatment group g at the occasion t on the original scale is given by

$$\xi_{(g,t)} = \left\{\lambda\left(\beta_I + \beta_{g(g)} + \beta_{t(t)} + \beta_{gt(g,t)} + \boldsymbol{x}_{\bar{c}}^T\boldsymbol{\beta}_c\right) + 1\right\}^{1/\lambda},$$

where $\boldsymbol{x}_{\bar{c}}$ is the vector of the mean of each covariate for all subjects. The model median is the inverse Box–Cox transformation of the model means on the Box–Cox transformed scale. The model median can be easily interpreted because it is the estimator of the median on the original scale.

Using the delta method, the variance estimator of the maximum likelihood estimator for the model median, $\hat{\xi}_{(g,t)}$, is given by $V_{\xi(g,t)}^{(\cdot)} = \Delta_{\xi(g,t)}^T V_{\boldsymbol{\theta}}^{(\cdot)} \Delta_{\xi(g,t)}$, where $\Delta_{\xi(g,t)} = \frac{\partial}{\partial\boldsymbol{\theta}}\xi_{(g,t)}\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$. If we use $V_{\boldsymbol{\theta}}^{(\cdot)} = V_{\boldsymbol{\theta}}^{(M)}$, we obtain the model-based variance estimator, $V_{\xi(g,t)}^{(M)}$. On the other hand, the robust variance estimator, $V^{(R)}_{xi(g,t)}$, is obtained if we use $V^{(\cdot)}_{\boldsymbol{\theta}} = V^{(R)}_{\boldsymbol{\theta}}$.

Thus, the model median difference between groups g_1 and g_2 at occasion t is given by $\delta_{(g_1;g_2,t)} = \xi_{(g_1,t)} - \xi_{(g_2,t)}$ and the variance estimators of the maximum likelihood estimator of the model median difference, $\hat{\delta}_{(g_1;g_2,t)}$, is given by $V_{\delta(g_1;g_2,t)} = (\Delta_{(g_1,t)} - \Delta_{(g_2,t)})^T V^{(\cdot)}_{\boldsymbol{\theta}} (\Delta_{(g_1,t)} - \Delta_{(g_2,t)})$. The model-based and robust variance estimators are obtained in the same way as the model median estimator. Using the asymptotic normality of the maximum likelihood estimator, the $100(1-\alpha)\%$ confidence interval of $\delta_{(g_1;g_2,t)}$ is obtained as $\hat{\delta}_{(g_1;g_2,t)} \pm \Phi^{-1}(1-\alpha/2)\sqrt{V^{(\cdot)}_{\delta(g_1;g_2,t)}}$, where $\Phi^{-1}(\cdot)$ is the quantile function of the standard normal distribution. The Wald-type test for the null hypothesis, $\delta_{(g_1;g_2,t)} = 0$, is performed with the test statistic $t = \hat{\delta}_{(g_1;g_2,t)} / \sqrt{V^{(\cdot)}_{\delta(g_1;g_2,t)}}$.

The performances of the above-mentioned inference procedures may be low for a small sample because these are based on the asymptotic properties of the maximum likelihood estimation. Therefore, Maruo et al. (2017) applied the following empirical small sample adjustment for the inferences of the model median differences by referring to the study in Schluchter and Elashoff (1990).

They provide an adjusted standard error (SE) for the median difference as $\sqrt{M/(M-p)V^{(\cdot)}_{\delta(g_1;g_2,t)}}$ for the compound symmetry (CS) or the first-order auto regression (AR(1)) structure and $\sqrt{n^*/(n^*-T)} \times \sqrt{V^{(\cdot)}_{\delta(g_1;g_2,t)}}$ for the unstructured (UN) structure. Further, we approximate the null distribution by the t distribution where the degrees of freedom for the CS or the AR(1) structure and the UN structure are $(n-G)(T-1)-m$ and n^*-T, respectively.

Although Maruo et al. (2017) and Maruo et al. (2020) focus only on the three covariance structures, these three options are sufficient in the applied settings because the MMRM analysis is often applied in the following steps. The UN structure is used, and the CS or AR(1) structure with the robust variance estimation is used when the parameter estimation process is not properly converged (e.g., see Gosho et al. (2017)).

Illustrations

In this section, we describe the **bcmixed** package that provides the analysis results based on the mixed effect models with the Box–Cox transformation. The package **bcmixed** is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=bcmixed. The package **bcmixed** can be installed and loaded using the following code.

```
R> install.packages("bcmixed")
R> library(bcmixed)
```

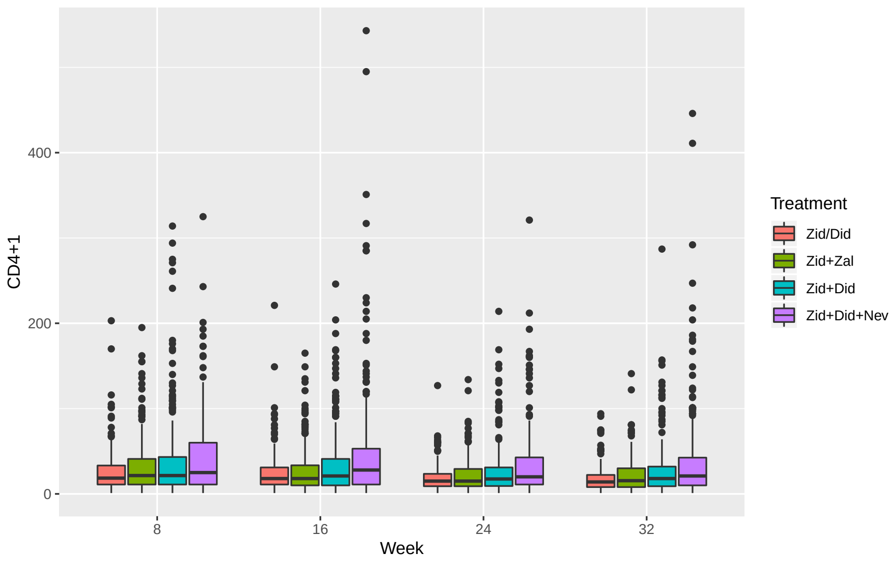


Figure 1: Longitudinal box-whisker plot of outcome variable (`cd4`) for each treatment group in the `aidscd4` dataset.

In particular, the following main functions are demonstrated in this article:

`bcmarg`	: Inference on the marginal model of the mixed effect model with the Box–Cox transformation.
`bcmmrm`	: Inference on the model median for longitudinal data in randomized clinical trials.

Example data

First, we illustrate a real example for the acquired immune deficiency syndrome (AIDS) clinical trial data, which is stored as the data frame `aidscd4` in the **bcmixed** package. The data are from a randomized, double-blind study of AIDS patients with advanced immune suppression (cluster of differentiation 4 (CD4) cells count of less than or equal to 50 cells/mm3) (Henry et al., 1998; Fitzmaurice et al., 2011). Patients in the AIDS clinical trial group study 193A were randomized to dual or triple combinations of human immunodeficiency virus-1 reverse transcriptase inhibitors. In particular, the patients were randomized to one of four daily regimens. The original data can be downloaded from `https://content.sph.harvard.edu/fitzmaur/ala/` (Datasets->AIDS Clinical Trial Group (ACTG) Study 193A). As for the more detailed data handling process, see Maruo et al. (2017). The data frame `aidscd4` contains seven variables (Table 1).

Variable	Description
`id`	A patient identifier; in total there are 1177 patients.
`weekc`	A visit variable (weeks 8, 16, 24, 32).
`treatment`	Allocated treatment regimens; 1 = zidovudine alternating monthly with 400mg didanosine, 2 = zidovudine plus 2.25mg of zalcitabine, 3 = zidovudine plus 400mg of didanosine, and 4 = zidovudine plus 400mg of didanosine plus 400mg of nevirapine.
`age`	Patients' age (years).
`sex`	Patients' sex (1 = male, `0` = female).
`cd4.bl`	A baseline value of CD4 cells count + 1.
`cd4`	A CD4 cells count + 1.

Table 1: Variable definition of AIDS clinical trial data

Figure 1 shows the longitudinal box-whisker plot of the values of CD4 plus 1 for each group plotted using **ggplot2** package (Wickham, 2016). The distribution shapes of the measurements were heavily skewed.

bcmarg function

This function provides the inference results for the marginal model of the mixed effect model with the Box–Cox transformation described in Section Parameter inference for the Box–Cox model.

Usage: The `bcmarg` function is called using the following syntax.

```
bcmarg(formula, data, time = NULL, id = NULL, structure = "UN",
  lmdint = c(-3, 3))
```

Arguments: The `bcmarg` function takes arguments tabulated in Table 2.

Argument	Description
`formula`	A two-sided linear formula object describing the model, with the response on the left of a ~ operator and the terms, separated by + operators, on the right.
`data`	A data frame containing the variables used in the model.
`time`	A time variable name for repeated measurements. The default is `NULL`.
`id`	A subject id variable name for repeated measurements. The default is `NULL`.
`structure`	Specify the covariance structure from `c("UN","CS","AR(1)")`. The default is `"UN"`.
`lmdint`	A vector containing the end points of the interval to be searched for a transformation parameter. The default is `c(-3,3)`.

Table 2: Auguments of bcmarg function

If time and id are not specified, an ordinary linear model with the Box–Cox transformation is applied.

Value: The bcmarg function returns an object of class "bcmarg". The objects of this class have methods for generic functions coef, logLik, vcov, fitted, print, and summary. The object includes the components for the marginal model parameter inference (Table 3).

Component	Description
lambda	A numeric value of the estimate of the transformation parameter.
beta	A vector with the estimates of the regression parameters.
alpha	A vector with the estimates of the covariance parameters.
V	A variance-covariance matrix for any subject with no missing values.
betainf	A matrix containing the inference results for beta under the assumption that lambda is known.
Vtheta.mod	A model-based variance-covariance matrix for MLE of the vector of all parameters: c(lambda,beta,alpha).
Vtheta.rob	A robust variance-covariance matrix for MLE of the vector of all parameters.
logLik	A numeric value of the maximized likelihood.
adj.prm	A vector with parameters used for the empirical small sample adjustment in bcmmrm: c(number of subjects, number of completed subjects, number of outcome observations, number of missing observations).
glsObject	An object of "gls" (or "lm" when time and id are not specified) containing results of gls (or lm) function on the transformed scale.

Table 3: Values of bcmarg function

In bcmarg function, lambda is estimated with the optimize function by maximizing the profile likelihood function for λ. If an error occurs in the optimize function, problems may be solved by changing the search area for λ, lmdint.

Example code: We applied a marginal model to the aidscd4 data, where the fixed effects were the treatment, visit, treatment-visit interaction, and the Box–Cox transformed baseline, where the visit was treated as a nominal variable. The covariance structure was set as unstructured (default setting). This model is frequently used for MMRM analysis. A sample code is as follows. The bct.v function returns the Box–Cox transformed vector.

```
R> data("aidscd4")
R> aidscd4$cd4.bl.tr <- bct.v(aidscd4$cd4.bl)$transformed
R> res1 <- bcmarg(formula = cd4 ~ as.factor(treatment) * as.factor(weekc) +
+                  cd4.bl.tr, data = aidscd4, time = weekc, id = id)
```

The summarized analysis results on the transformed scale are obtained by applying the summary function to the "bcmarg" object as follows.

```
R> summary(res1)

Box-Cox transformed mixed model analysis
  Formula: cd4 ~ as.factor(treatment) * as.factor(weekc) + cd4.bl.tr
  Time: weekc
  ID: id
  Covariance structure: "UN"
  Data: aidscd4
  Log-likelihood: -13322.96
  Estimated transformation parameter:  0.154

Coefficients on the transformed scale:
                                         Value Std.Error t-value p-value
(Intercept)                             1.0849    0.1249   8.684   0.000
as.factor(treatment)2                   0.2454    0.1214   2.022   0.043
as.factor(treatment)3                    0.4203    0.1212   3.468   0.001
as.factor(treatment)4                    0.7649    0.1199   6.377   0.000
as.factor(weekc)16                      -0.2043    0.0843  -2.424   0.015
as.factor(weekc)24                      -0.4498    0.0899  -5.004   0.000
as.factor(weekc)32                      -0.6750    0.0988  -6.835   0.000
cd4.bl.tr                                0.5782    0.0200  28.922   0.000
as.factor(treatment)2:as.factor(weekc)16 -0.1183    0.1185  -0.998   0.318
as.factor(treatment)3:as.factor(weekc)16 -0.0560    0.1180  -0.475   0.635
as.factor(treatment)4:as.factor(weekc)16  0.0675    0.1175   0.574   0.566
as.factor(treatment)2:as.factor(weekc)24 -0.1863    0.1264  -1.474   0.141
as.factor(treatment)3:as.factor(weekc)24 -0.0243    0.1264  -0.193   0.847
as.factor(treatment)4:as.factor(weekc)24  0.0870    0.1263   0.689   0.491
as.factor(treatment)2:as.factor(weekc)32 -0.0852    0.1414  -0.603   0.547
as.factor(treatment)3:as.factor(weekc)32 -0.0893    0.1400  -0.638   0.524
as.factor(treatment)4:as.factor(weekc)32  0.1414    0.1381   1.024   0.306

Covariance parameters on the transformed scale:
UN(1,1) UN(1,2) UN(1,3) UN(1,4) UN(2,2) UN(2,3) UN(2,4) UN(3,3) UN(3,4) UN(4,4)
  1.798   1.156   1.105   0.927   1.957   1.346   1.298   1.903   1.452   2.009

Correlations on the transformed scale:
       8    16    24    32
8  1.000 0.616 0.598 0.488
16 0.616 1.000 0.697 0.655
24 0.598 0.697 1.000 0.743
32 0.488 0.655 0.743 1.000
```

The results of coefficients on the transformed scale are obtained with the gls function in the **nlme** package. The transformation parameter was estimated as 0.154, which suggested that the shape of the residual distribution was close to a multivariate log-normal distribution. All main effects were significant; however, treatment-by-week interaction was not significant at a level of 0.05. Note that inference results for beta under the assumption that the transformation parameter is known are provided. Although statistical tests would be asymptotically valid (e.g., see Doksum and Wong (1983)), standard errors might be underestimated (e.g., see Bickel and Doksum (1981)).

bcmmrm function

This function provides the results for the model median inferences for longitudinal randomized clinical trial data described in Section Model median inference. The parameter inference is conducted by calling the bcmarg function in the bcmmrm function.

Usage: The bcmmrm function is called using the following syntax.

```
bcmmrm(outcome, group, data, time = NULL, id = NULL, covv = NULL,
  cfactor = NULL, structure = "UN", lmdint = c(-3, 3), glabel = NULL,
  tlabel = NULL)
```

Argument: The bcmmrm function takes arguments tablated in Table 4.

Argument	Description
outcome	A name of positive outcome (dependent) variable included in data.
group	A name of treatment group variable included in data.
data	A data frame that may include outcome, group, time, id, and specified covariate variables.
time	A name of time variable for repeated measurements included in data. The default is NULL.
id	A name of subject id variable for repeated measurements included in data. The default is NULL.
covv	A character vector with names of covariate variables included in data. The default is NULL.
cfactor	An integer vector including nominal variable indicators for covariate variables. Nominal variable: 1, continuous variable: 0. The default is NULL.

`structure`	Specify the covariance structure from `c("UN","CS","AR(1)")`. The default is `"UN"`.
`conf.level`	A numeric value of the confidence level for the confidence intervals. The default is 0.95.
`lmdint`	A vector containing end points of the interval to be searched for a transformation parameter. The default is `c(-3,3)`.
`glabel`	A vector of length number of treatment groups containing the labels of `group` variable. The default is `NULL` and the levels of `group` variable in `data` are used.
`tlabel`	A vector of length number of repeated measures containing the labels of `time` variable. The default is `NULL` and the levels of `time` variable in `data` are used.

Table 4: Auguments of `bcmmrm` function

If `time` and `id` are not specified, inference results reduce to the results for the context of linear regression model provided by Maruo et al. (2015).

Value: The `bcmmrm` function returns an object of class `"bcmmrm"` representing the results of model median inference based on the Box–Cox transformed MMRM analysis. Generic functions `boxplot`, `coef`, `logLik`, `vcov`, `fitted`, `plot`, `print`, and `summary` have methods to demonstrate the results of the fit. Components tablated in Table 5 must be included in a legitimate `"bcmmrm"` object.

Component	Description
`call`	A list containing an image of the `bcmmrm` call that produced the object.
`median.mod`, `median.rob`, `median.mod.adj`, `median.rob.adj`	Lists including inference results for the model medians for all groups. Levels of the list are time points, where the correspondence table is given as `time.tbl$code`. mod: model-based inference, rob: robust inference, adj: with empirical small sample adjustment.
`meddif.mod`, `meddif.rob`, `meddif.mod.adj`, `meddif.rob.adj`	Lists including inference results for the for the model median differences between all pairs of groups (group1 - group0). The levels of the list are time points, where the correspondence table is given as `time.tbl$code`. `mod`: model-based inference, `rob`: robust inference, `adj`: with empirical small sample adjustment.
`lambda`	A numeric value of estimates of the transformation parameter.
`R`	A correlation matrix for transformed outcomes.
`betainf`	Inference results for `beta` under the assumption that `lambda` is known.
`time.tbl`	A data frame of a correspondence table for the time points. This object is used when applying the above generic functions.
`group.tbl`	A data frame of a correspondence table for treatment groups. This object is used when applying the above generic functions.
`inf.marg`	A `"bcmarg"` object with results of the `bcmarg` function called in the `bcmmrm` function.
`outdata`	A data frame where the transformed outcome (`ytr`), the fitted values on the transformed scale (`ytr.fitted`), and the residuals on the transformed scale (`res.tr`) are added to input data.
`conf.level`	A numeric value of the specified confidence level.

Table 5: Values of `bcmmrm` function

`lambda`, `R`, `betainf`, and `inf.marg` are obtained from the results of the `bcmarg` function that is called in the `bcmmrm` function.

Sample code: We applied the MMRM analysis with the Box–Cox transformation described in Section bcmarg function to the `aidscd4` data, where the Box–Cox transformed baseline and sex were included as the covariates. The example code is as follows.

```
R> data("aidscd4")
R> aidscd4$cd4.bl.tr <- bct.v(aidscd4$cd4.bl)$transformed
R> res2 <- bcmmrm(outcome = cd4, group = treatment, data = aidscd4, time = weekc,
+                 id = id, covv = c("cd4.bl.tr", "sex"),  cfactor = c(0, 1),
+                 glabel = c("Zid/Did", "Zid+Zal", "Zid+Did", "Zid+Did+Nev")
```

The transformed baseline and sex are continuous and categorical variables, respectively, and therefore, `cfactor` was set as `c(0,1)`.

The print function only provides information about model detail, the estimated transformation parameter, the maximized log-likelihood, and the model median estimate for each time point and group as follows.

```
R> print(res2)

Model median estimation based on MMRM with Box-Cox transformation
  Outcome: cd4
  Group: treatment
  Time: weekc
  ID: id
  Covariate(s): c("cd4.bl.tr", "sex")
  Covariance structure: "UN"
  Data: aidscd4
  Estimated transformation parameter:  0.154
  Log-likelihood: -13322.36

Model median estimates (row: group, col: time):
  treatment | weekc    8   16   24   32
1           Zid/Did 18.9 16.5 14.1 12.1
2           Zid+Zal 22.0 17.9 14.6 13.5
3           Zid+Did 24.5 20.9 18.2 15.1
4       Zid+Did+Nev 30.1 27.8 24.2 21.8
```

The summary function provides more detailed analysis results as follows.

```
R> summary(res2)

Model median inference based on MMRM with the Box-Cox transformation

Data and variable information:
  Outcome: cd4
  Group: treatment
  Time: weekc
  ID: id
  Covariate(s): c("cd4.bl.tr", "sex")
  Data: aidscd4

Analysis information:
  Covariance structure: "UN"
  Robust inference: TRUE
  Empirical small sample adjustment: TRUE
  Confidence level: 0.95

Analysis results:
  Estimated transformation parameter:  0.154


Model median inferences for weekc = 8

    treatment median    SE lower.CL upper.CL
1     Zid/Did   18.9 0.862     17.2     20.6
2     Zid+Zal   22.0 1.124     19.8     24.2
3     Zid+Did   24.5 1.465     21.6     27.4
4 Zid+Did+Nev   30.1 1.597     27.0     33.3

(...Omitted for weeks 16 and 24...)

Model median inferences for weekc = 32

    treatment median    SE lower.CL upper.CL
1     Zid/Did   12.1 0.662     10.8     13.4
2     Zid+Zal   13.5 0.813     11.9     15.1
3     Zid+Did   15.1 1.019     13.1     17.1
4 Zid+Did+Nev   21.8 1.376     19.1     24.5
```

```
Inferences of model median difference between groups
  ( treatment 1 - treatment 0 ) for weekc = 8

  treatment 1 treatment 0 delta   SE lower.CL upper.CL t.value p.value
1     Zid+Zal     Zid/Did  3.12 1.40    0.363     5.87    2.22   0.027
2     Zid+Did     Zid/Did  5.64 1.69    2.325     8.96    3.34   0.001
3 Zid+Did+Nev     Zid/Did 11.25 1.80    7.711    14.80    6.24   0.000
4     Zid+Did     Zid+Zal  2.53 1.83   -1.059     6.12    1.39   0.167
5 Zid+Did+Nev     Zid+Zal  8.14 1.93    4.349    11.93    4.22   0.000
6 Zid+Did+Nev     Zid+Did  5.61 2.16    1.372     9.85    2.60   0.010

(...Omitted for weeks 16 and 24...)

Inferences of model median difference between groups
  ( treatment 1 - treatment 0 ) for weekc = 32

  treatment 1 treatment 0 delta   SE lower.CL upper.CL t.value p.value
1     Zid+Zal     Zid/Did  1.35 1.04   -0.692     3.40    1.30   0.194
2     Zid+Did     Zid/Did  3.00 1.20    0.633     5.37    2.49   0.013
3 Zid+Did+Nev     Zid/Did  9.70 1.52    6.705    12.69    6.37   0.000
4     Zid+Did     Zid+Zal  1.65 1.30   -0.907     4.20    1.27   0.206
5 Zid+Did+Nev     Zid+Zal  8.34 1.60    5.206    11.48    5.23   0.000
6 Zid+Did+Nev     Zid+Did  6.70 1.71    3.338    10.06    3.92   0.000
```

Significant differences were detected for the following pairs Zid+Did vs. Zid/Did, Zid+Did+Nev vs. Zid/Did, Zid+Did+Nev vs. Zid+Zal, and Zid+Did+Nev vs. Zid+Did at week 32.

The `summary` function provides the results using the robust variance and the small sample adjustment in the default settings. If users want to summarize results using the model variance and not using the small sample adjustment, specify `summary(bcmmrmObject,robust = F,ssadjust = F)`. Further details of the `summary` function for the `"bcmmrm"` object can be obtained with `?summary.bcmmrm`.

The inference results for the median differences at week 32 (fourth visit) can also be called as follows although the levels of the group variable in the data frame are used without formatting.

```
R> res2$meddif.rob.adj[[4]]

  group1 group0    delta       SE   lower.CL  upper.CL  t.value      p.value
1      2      1 1.354438 1.041338 -0.6922404  3.401117 1.300672 1.940595e-01
2      3      1 3.000942 1.204919  0.6327547  5.369129 2.490575 1.312570e-02
3      4      1 9.697631 1.522831  6.7046091 12.690653 6.368159 4.869820e-10
4      3      2 1.646503 1.299231 -0.9070469  4.200054 1.267291 2.057293e-01
5      4      2 8.343192 1.596236  5.2058987 11.480486 5.226792 2.682862e-07
6      4      3 6.696689 1.709027  3.3377114 10.055667 3.918421 1.034880e-04
```

The `"bcmmrm"` object can be plotted with the `plot` function as follows (Figure 2).

```
R> plot(res2, xlab = "Week", ylab = "CD4+1")
```

The `plot` function provides a longitudinal plot in the default settings. However, a plot at a specified time point can be drawn with the following code (Figure 3):

```
R> plot(res2, timepoint = 32, xlab = "Treatment", ylab = "CD4+1", col = 1:4)
```

Further, the `plot` function provides the results using the robust variance and the small sample adjustment in the default setting. Many other options such as `main` and `legend` can be used in the `plot` function. Further details can be obtained with `?plot.bcmmrm`.

A diagnosis of a model fitting can be conducted with the `boxplot` function, which provides a box-whisker plot of the Box–Cox transformed residuals for each group. A sample code is provided as follows (Figure 4):

```
R> boxplot(res2)
```

The shape of the transformed residual for each group is not skewed and the median and mean are close to each other, which suggests that the median would not be biased at week 32. The `boxplot` function provides the results at the last time points in the default settings. A box-whisker plot at another time point can be obtained by specifying `boxplot(bcmmrmObject,timepoint = xx)`.

Summary and discussion

We demonstrated the use of the **bcmixed** package. The bcmarg function provides the analysis results for a marginal model of a mixed effect model with the Box–Cox transformation. The results for the statistical test of the bcmarg function are meaningful. However, it is difficult to interpret coefficients (β) on the transformed scale.

The bcmmrm function provides the model median inferences based on the MMRM with the Box–Cox transformation for longitudinal randomized clinical trials. Using this function, treatment effects can be interpreted as median differences between treatment groups at specified time points. Maruo et al.

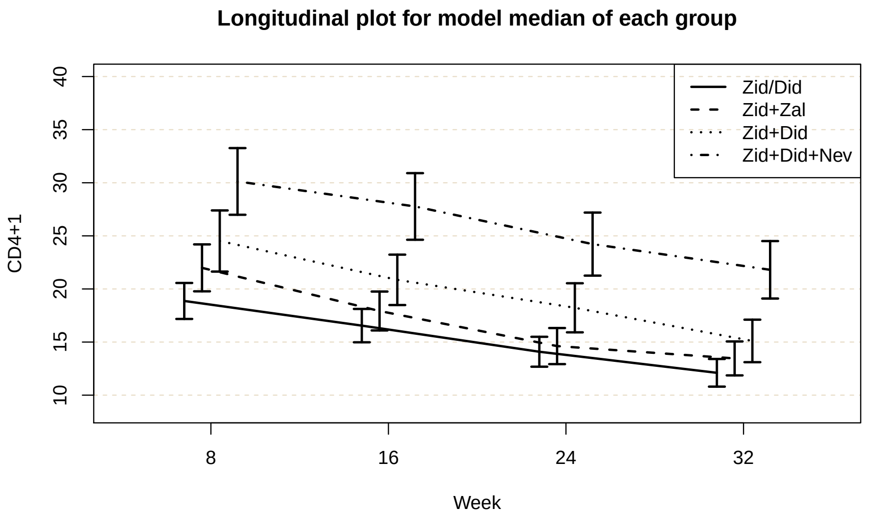


Figure 2: Longitudinal plot of model median for each group, created by applying plot function to bcmmrm object.

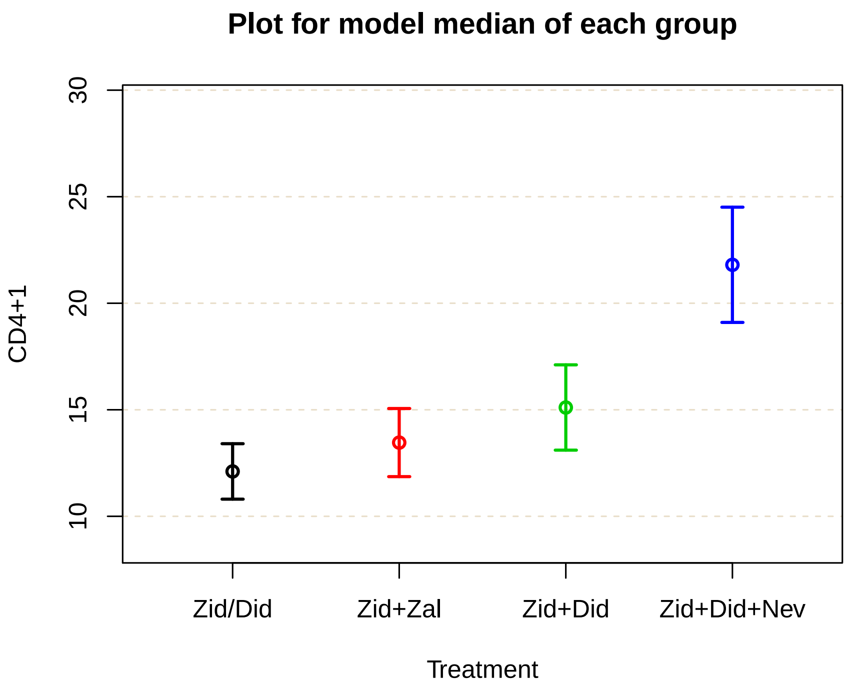


Figure 3: Plot of model median for each group at week 32, created by applying plot function to bcmmrm object with timepoint option.

(2017) also show that this method controls the type I error of the statistical test for the model median difference, and it has moderate or high performance for power compared with the existing methods (ordinary MMRM, MMRM for the log-transformed outcome, etc.) from their simulation studies (the simulation program is provided in `https://github.com/kzkzmr/Maruo_2017_StatMed_Simulation` with penalized power results proposed by Cavus et al. (2019)). Thus, `bcmmrm` function analysis results with high power and high interpretability for longitudinal randomized clinical trials with skewed outcomes. Further, the `bcmmrm` function provides summarizing and visualization tools, which would be helpful to write clinical trial reports.

Although the **bcmixed** package can be used for data other than that of randomized clinical trials, the performances of these methods have not been evaluated well yet. Therefore, users should use them carefully. Although a model fitting can be diagnosed with the `boxplot` function, more model diagnosis tools may be implemented in the future. Users might apply the **MissMech** package(Jamshidian et al., 2014, 2015), which diagnoses multivariate normality and heteroscedasticity, to the transformed residuals stored in `"bcmarg"` object. Note that the **MissMech** package assumes missing mechanisms are missing completely at random (MCAR), and statistical tests for model fittings may lead to significant results for a medium-to-large sample even when the models fit the data adequately.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP19K11849.

Bibliography

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01. [p253]

P. J. Bickel and K. A. Doksum. An analysis of transformations revisited. *Journal of the American Statistical Association*, 76:296–311, 1981. URL `https://doi.org/10.1080/01621459.1981.10477649`. [p258]

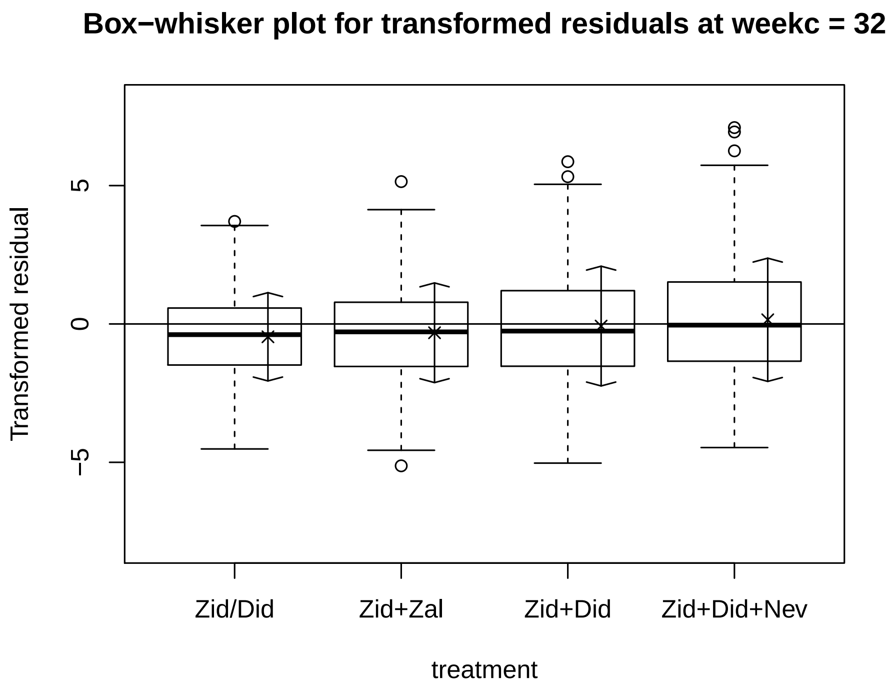


Figure 4: Box-whisker plot of transformed residuals for each group at week 32, created by applying `boxplot` function to `bcmmrm` object.

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journals of the Royal Statistical Society Series B*, 26:211–246, 1964. URL https://doi.org/10.1111/j.2517-6161.1964.tb00553.x. [p253, 254]

M. Cavus, B. Yazici, and A. Sezer. Penalized power approach to compare the power of the tests when type i error probabilities are different. *Communications in Statistics - Simulation and Computation*, 0(0): 1–15, 2019. doi: 10.1080/03610918.2019.1588310. [p263]

D. R. Cox. Tests of separate families of hypotheses. In *Proceeding of 4th Berkley Symposium 1*, pages 105–123, California, 1961. University of California Press. [p254]

K. A. Doksum and C. W. Wong. Statistical tests based on transformed data. *Journal of the American Statistical Association*, 78:411–417, 1983. URL https://doi.org/10.1080/01621459.1983.10477986. [p258]

G. M. Fitzmaurice, N. M. Laird, and J. H. Ware. *Applied Longitudinal Analysis, 2nd Edition*. Wiley, New York, 2011. URL https://doi.org/10.1002/9781119513469. [p256]

M. Gosho, A. Hirakawa, H. Noma, K. Maruo, and Y. Sato. Comparison of bias-corrected covariance estimators for mmrm analysis in longitudinal data with dropouts. *Statistical Methods in Medical Research*, 26:2389–2406, 2017. URL https://doi.org/10.1177/0962280215597938. [p255]

J. D. Hadfield. Mcmc methods for multi-response generalized linear mixed models: The MCMCglmm R package. *Journal of Statistical Software*, 33(2):1–22, 2010. URL https://www.jstatsoft.org/v33/i02/. [p253]

K. Henry, A. Erice, C. Tierney, H. H. Balfour Jr., M. A. Fischl, A. Kmack, S. H. Liou, A. Kenton, M. S. Hirsch, J. Phair, A. Martinez, and J. O. Kahn for the AIDS Clinical Trial Group 193A Study Team. A randomized, controlled, double-blind study comparing the survival benefit of four different reverse transcriptase inhibitor therapies (three-drug, two-drug, and alternating drug) for the treatment of advanced AIDS. *Journal of Acquired Immune Deficiency Syndromes and Human Retrovirology*, 19: 339–349, 1998. URL https://doi.org/10.1097/00042560-199812010-00004. [p256]

M. Jamshidian, S. Jalal, and C. Jansen. **MissMech**: an r package for testing homoscedasticity, multivariate normality, and missing completely at random (mcar). *Journal of Statistical Software*, 56(6), 2014. URL https://doi.org/10.18637/jss.v056.i06. [p263]

M. Jamshidian, S. Jalal, and C. Jansen. *Package '**MissMech**:'*, 2015. URL https://CRAN.R-project.org/package=MissMech. R package version 1.0.2. [p263]

C. M. Jelsema and S. D. Peddada. CLME: An R package for linear mixed effects models under inequality constraints. *Journal of Statistical Software*, 75(1):1–32, 2016. doi: 10.18637/jss.v075.i01. [p253]

S. R. Lipsitz, J. Ibrahim, and G. Molenberghs. Using a Box–Cox transformation in the analysis of longitudinal data with incomplete responses. *Journal of the Royal Statistical Society, Series C*, 49: 287–296, 2000. URL https://doi.org/10.1111/1467-9876.00192. [p253, 254]

C. H. Mallinckrodt, W. S. Clark, and S. R. David. Accounting for dropout bias using mixed-effects models. *Journal of Biopharmaceutical Statistics*, 11:9–21, 2001. URL https://doi.org/10.1081/BIP-100104194. [p253]

K. Maruo, N. Isogawa, and M. Gosho. Inference of median difference based on the Box–Cox model in randomized clinical trials. *Statistics in Medicine*, 34:1634–1644, 2015. URL https://doi.org/10.1002/sim.6408. [p253, 259]

K. Maruo, N. Yamaguchi, H. Noma, and M. Gosho. Interpretable inference on the mixed effect model with the Box–Cox transformation. *Statistics in Medicine*, 36:2420–2434, 2017. URL https://doi.org/10.1002/sim.7279. [p253, 254, 255, 256, 262]

K. Maruo, R. Ishii, Y. Yamaguchi, and M. Gosho. *Package '**bcmixed**'*, 2020. URL https://CRAN.R-project.org/package=bcmixed. R package version 1.0. [p253, 255]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2021. URL https://CRAN.R-project.org/package=nlme. R package version 3.1-152. [p253]

N. Rockwood and M. Jeon. Estimating complex measurement and growth models using the R package PLmixed. *Multivariate Behavioral Research*, 54(2):288–306, 2019. [p253]

Sam Weerahandi, Berna Yazici, Ching-Ray Yu, and Mustafa Cavus. *glme: Generalized Linear Mixed Effects Models*, 2021. URL https://CRAN.R-project.org/package=glme. R package version 0.1.0. [p253]

M. D. Schluchter and J. D. Elashoff. Small-sample adjustments to tests with unbalanced repeated measures assuming several covariance structures. *Journal of Statistical Computation and Simulation*, 37:69–87, 1990. URL https://doi.org/10.1080/00949659008811295. [p255]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p256]

Kazushi Maruo
Department of Biostatistisc, Faculty of Medicine, University of Tsukuba
1-1-1, Tennodai, Tuskuba, Ibaraki,
Japan
ORCiD: 0000-0002-3093-0386
kazushi.maruo@gmail.com

Ryota Ishii
Biostatistics Unit, Clinical and Translational Research Center, Keio University Hospital
(Present affiliation: Department of Biostatistisc, Faculty of Medicine, University of Tsukuba
1-1-1, Tennodai, Tuskuba, Ibaraki,
Japan)
rishii@md.tsukuba.ac.jp

Yusuke Yamaguchi
Data Science, Development, Astellas Pharma Inc.
2-5-1, Nihonbashi-Honcho, Chuo-Ku, Tokyo,
Japan
yusuke-yamaguchi@astellas.com

Masahiko Gosho
Department of Biostatistisc, Faculty of Medicine, University of Tsukuba
1-1-1, Tennodai, Tuskuba, Ibaraki,
Japan
mgosho@md.tsukuba.ac.jp

diproperm: An R Package for the DiProPerm Test

by Andrew G. Allmon, J.S. Marron, and Michael G. Hudgens

Abstract High-dimensional low sample size (HDLSS) data sets frequently emerge in many biomedical applications. The direction-projection-permutation (DiProPerm) test is a two-sample hypothesis test for comparing two high-dimensional distributions. The DiProPerm test is exact, i.e., the type I error is guaranteed to be controlled at the nominal level for any sample size, and thus is applicable in the HDLSS setting. This paper discusses the key components of the DiProPerm test, introduces the **diproperm** R package, and demonstrates the package on a real-world data set.

Introduction

Advancements in modern technology and computer software have dramatically increased the demand and feasibility to collect high-dimensional data sets. Such data introduce analytical challenges which require the creation of new and adaptation of existing statistical methods. One such challenge is that we may observe many more covariates (features) p than the number of observations N, especially in small sample size studies. These data structures are known as high-dimensional, low sample size (HDLSS) data sets, or "small N, big p."

HDLSS data frequently occur in many health-related fields. For example, in genomic studies, a single microarray experiment can produce tens of thousands of gene expressions compared to the few samples studied, often being less than a hundred (Alag, 2019). In medical imaging studies, a single region of interest for analysis in an MRI or CT-scan image often contains thousands of features compared to the small number of samples studied (Limkin et al., 2017). In the pre-clinical evaluation of vaccines and other experimental therapeutic agents, the number of biomarkers measured (e.g., immune responses) may be much greater than the number of samples studied (e.g., mice, rabbits, or non-human primates) (Kimball et al., 2018).

One common task in the HDLSS setting entails comparing the distribution of covariates between two groups (classes). For example, in pre-clinical studies of vaccines, investigators may wish to compare the distribution of biomarkers between animals who survive to a certain time point and animals who do not survive. In the example presented below, the distribution of $p = 112$ covariates is compared between edible and poisonous mushrooms. The covariate distributions may be compared between two samples using the direction-projection-permutation (DiProPerm) test (Wei et al., 2016). This test is well-suited for the HDLSS setting because the test is exact, i.e., the type I error is guaranteed to be controlled at the nominal level for any sample size. Below we discuss the key components of the DiProPerm test, introduce the **diproperm** R package, and demonstrate the use of the package on a real-world data set.

DiProPerm Test

To compare the distribution of covariates between two groups, the DiProPerm tests use one-dimensional projections of the data based on a binary linear classifier to construct a univariate test statistic and then permutes class labels to determine the sampling distribution of the test statistic under the null. The details of the DiProPerm test are as follows.

Let $X_1, \ldots, X_n \sim F_1$ and $Y_1, \ldots, Y_m \sim F_2$ be independent random samples of p dimensional random vectors from multivariate distributions F_1 and F_2 where p may be larger than m and n. The DiProPerm test evaluates the hypotheses

$$H_0 : F_1 = F_2 \text{ versus } H_1 : F_1 \neq F_2$$

The test entails three steps:

1. Direction: find the normal vector to the separating hyperplane between two samples after training a binary linear classifier
2. Projection: project data on to the normal vector and calculate a univariate two-sample statistic
3. Permutation: conduct a permutation test using the univariate statistic as follows:
   (a) Permute class membership after pooling samples

(b) Re-train binary classifier and find the normal vector to the separating hyperplane

(c) Recalculate the univariate two sample statistic

(d) Repeat a-c multiple times (say 1000) to determine the sampling distribution of the test statistic under the null H_0

(e) Compute p-value by comparing the observed statistic to the sampling distribution

Different binary linear classifiers may be used in the first step of DiProPerm. Linear discriminant analysis (LDA), particularly after conducting principal component analysis (PCA), is one possible classifier for the direction step. However, using LDA with PCA in the HDLSS setting has some disadvantages, including lack of interpretability, sensitivity to outliers, and tendency to find spurious linear combinations due to a phenomenon known as data piling (Aoshima et al., 2018; Marron et al., 2007). Data piling occurs if data are projected onto some projection direction, and many of the projections are the same or piled on one another. The support vector machine (SVM) is another popular classifier (Hastie et al., 2001). The SVM finds the hyperplane that maximizes the minimum distance between data points and the separating hyperplane. However, the SVM can also suffer from data piling in the HDLSS setting. To overcome data piling, the distance weighted discrimination (DWD) classifier was developed (Marron et al., 2007). The DWD classifier finds the separating hyperplane minimizing the average inverse distance between data points and the hyperplane. The DWD performs well in HDLSS settings with good separation and is more robust to data piling.

In the second step of DiProPerm, a univariate statistic is calculated using the projected values onto the normal vector to the separating hyperplane from the first step. Suppose $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ are the projected values from samples $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$, respectively. One common choice for the univariate test statistic for DiProPerm includes the difference of means statistic, $|\bar{x} - \bar{y}|$. Other two-sample univariate statistics such as the two-sample t-statistic or difference in medians are also possible for use with the DiProPerm.

The last step of DiProPerm entails determining the distribution of the test statistic under the null. In this step, the two samples are pooled, class labels are permuted, then a univariate statistic is calculated. Repeat this process multiple times (say 1000) to determine the sampling distribution of the test statistic under the null H_0. P-values are then calculated by the proportion of statistics higher than the original value.

When the DiProPerm test is implemented using the DWD classifier, it is common practice to look at the loadings of the DWD classifier (An et al., 2016; Nelson et al., 2019). The DWD loadings represent the relative contribution of each variable to the class difference. A higher absolute value of a variable's loading indicates a greater contribution for that variable to the class difference. The loadings vector is a unit vector; thus, the individual loadings range from -1 to 1, and the sum of the squares of the loadings equals one. The loadings direction vector points from the negative to positive class. Thus, positive entries correspond to variables that tend to be larger for the positive class. Combining the use of the DiProPerm and evaluation of the DWD loadings in applications can provide insights into high-dimensional data and be used to generate rational hypotheses for future research.

The DiProPerm test has been used in several areas of biomedical research, including osteoarthritis and neuroscience (An et al., 2016; Bendich et al., 2016; Nelson et al., 2019). However, currently, there does not exist an R package that implements DiProPerm. Therefore we developed **diproperm**, a free, publicly available R software package to analyze data from two high-dimensional distributions (Allmon et al., 2020). Functions in the **diproperm** package allow users to conduct the DiProPerm test and create corresponding diagnostic plots. Loadings for the binary linear classifier are also available for display in order from highest to lowest relative to their contribution toward the separation of the two distributions.

Package Overview

The **diproperm** package is comprised of three functions:

- `DiProPerm()`: Conducts the DiProPerm test
- `plotdpp()`: Plots diagnostics from the DiProPerm test
- `loadings()`: Returns the variable indices with the highest loadings in the binary classification. The absolute values of the loading values indicate a variable's relative contribution toward the separation between the two classes. The loadings vector is a unit vector, thus, the sum of its squares must be equal to one and range from -1 to 1. Also, the loadings direction vector points from the negative to positive class. Thus, positive entries correspond to variables that are larger for the positive class.

The **diproperm** package can be installed from CRAN using the code in the 'Example' section below. Additionally, the development version of the package can be installed from GitHub ("allmondrew/diproperm").

Example

The example below creates a Gaussian data set containing 100 samples, 2 features, clustered around 2 centers with a standard deviation of 2. The class labels are then re-classified to -1 and 1 to match the input requirements of `DiProPerm()`. The DiProPerm test is then conducted using the DWD classifier, the mean difference univariate statistic, and 1000 permutations. The results from `DiProPerm()` are then displayed with `plotdpp()`. Last, the top five indices of the highest absolute loadings are listed.

```
install.packages("diproperm")
remotes::install_github("elbamos/clusteringdatasets")
library(diproperm)
library(clusteringdatasets)

cluster.data <- make_blobs(n_samples = 100, n_features = 2, centers = 2, cluster_std = 2)

X <- cluster.data[[1]]
y <- cluster.data[[2]]
y[y==2] <- -1

dpp <- DiProPerm(X,y,B=1000,classifier = "dwd",univ.stat = "md")

plotdpp(dpp)

loadings(dpp,loadnum = 5)
```

Description

The main function to be called first by the user is `DiProPerm()`, which takes in an $n \times p$ data matrix and a vector of n binary class labels both provided by the user. Factor variables for the data matrix must be coded as 0/1 dummy variables, and the class labels for the vector of binary class labels must be coded as -1 and 1. By default, the `DiProPerm()` uses the DWD classifier, the mean difference as the univariate statistics, and 1000 balanced permutations. The permutations are balanced in the sense that after relabeling, the new -1 group contains $n/2$ members from the original -1 group and $n/2$ members not from the original -1 group. Users can also implement an unbalanced, randomized permutation design if desired. `DiProPerm()` implements DWD from the `genDWD` function in the **DWDLargeR** package (Lam et al., 2018a,b). The penalty parameter, `C`, in the `genDWD` function is calculated using the `penaltyParameter` function in **DWDLargeR**. **DWDLargeR** has several parameters which are set to recommended default values. More details on the algorithm used to compute `genDWD` and `penaltyParameter` can be found in Lam et al. (2018b). Other options included in `DiProPerm()` for the binary linear classifier are the mean difference direction "md" and support vector machine "svm". `DiProPerm()` uses parallel processing to delegate computation to the number of cores on the user's computer for increased efficiency. `DiProPerm()` returns a list of the observed data matrix, vector of observed class labels, observed test statistic, projection scores used to compute the observed test statistic, the loadings of the binary classification, the z-score, cutoff value for an α level of significance, the p-value for the DiProPerm test, a list of each permutation's projection scores and permuted class labels, and a vector of permuted test statistics the size of the number of permutations used.

After calling `DiProPerm()`, the function `plotdpp()` can be used to create a panel plot for assessing the diagnostics of the DiProPerm test. `plotdpp()` takes in a DiProPerm list and the user may specify which diagnostics they would like to display. By default, `plotdpp()` displays a facet plot with the observed score distribution, the projection score distribution of the permutation with the smallest test statistic value, the projection score distribution of the permutation with the largest test statistic value, and the test statistic permutation distribution. For the permutation distribution plot, the z-score, cutoff value, observed test statistic and p-value are displayed on the graph. Larger individual graphs may be displayed by using the plots option in `plotdpp()`. Additional graphs include the projection score distributions for the first and second permutations. The diagnostic plots show the user the characteristics of their data and facilitate the visual assessment of the separation of the two high-dimensional distributions being tested.

Lastly, after calling the `DiProPerm()`, the user may call the `loadings()` function. The `loadings()` function returns the variable indices in the data matrix which have the highest absolute loadings in the binary classification. Because the loadings vector is a unit vector, the sum of the squares of loadings is constrained to equal to one, with each loading between -1 to 1. The loadings direction vector points from the negative to the positive class. Thus, positive entries correspond to variables that tend to be larger for the positive class. Higher absolute loading values indicate a greater contribution for a particular variable toward the separation between the two classes. By default, `loadings()` returns the indices for all variables sorted by their absolute loading value. Therefore, the top variable index is the variable which contributes the most toward the separation of the two classes, and the last variable is the one which contributes the least. The user may also change the number of loadings displayed.

Application

To illustrate the use of the **diproperm** package, consider the mushrooms data set, which is freely available from the UCI Machine Learning Repository (Dua and Graff, 2019) and within **diproperm**. This data set includes various characterizations of 23 species of gilled mushrooms in the Agaricus and Lepiota families. Each mushroom species is labeled as either definitely edible or poisonous/unknown. There are $n = 8124$ mushrooms in total, and $p = 112$ binary covariates coded as 0/1 corresponding to 22 categorical attributes. Below, we demonstrate the **diproperm** package functionality using data from the first $n = 50$ mushrooms in the data set.

Step 1: Load and clean the data

```
install.packages("diproperm")
library(diproperm)
data(mushrooms)
```

The above code installs the **diproperm** package and loads the mushroom data into R. Now, let us check the structure of the data to make sure it is compatible with `DiProPerm()`.

```
dim(mushrooms$X)
    [1] 112 8124

table(mushrooms$y)
    -1 1
    4208 3916
```

The vector of class labels must be -1 or 1 for `DiProPerm()`, which is the case for this data. However, the data set is in $p \times n$ format. For `DiProPerm()`, the dataset must be in $n \times p$ format. This can be done using the transpose function from the **Matrix** package in R (Bates and Maechler, 2019). After taking the transpose, we subset the data and vector of class labels to the first 50 observations and store the results.

```
X <- Matrix::t(mushrooms$X)
X <- X[1:50,]
y <- mushrooms$y[1:50]
```

Step 2: Conduct DiProPerm

Now, that the data is in the proper format the call to `DiProPerm()` is as follows:

```
dpp <- DiProPerm(X=X,y=y,B=1000)

    Algorithm stopped with error 2.35e-08
    sample size =  50, feature dimension = 112
    positve sample =  12, negative sample =  38
    number of iterations =  51
    time taken = 0.10
```

Characteristics of the DWD algorithm used to find the solution for the observed data are displayed by `DiProPerm()`. The algorithm took 51 iterations and 0.10 seconds to converge to the tolerance threshold with a zero percent classification error on the training data set. The runtime for 1000 permutations was less than 3 minutes on a four-core machine but would be faster on a machine with more cores. The `dpp` object stores the output list from `DiProPerm()` described in the package. Storing the information allows us to plot the diagnostics in the next step.

Step 3: Plot diagnostics

```
plotdpp(dpp)
```

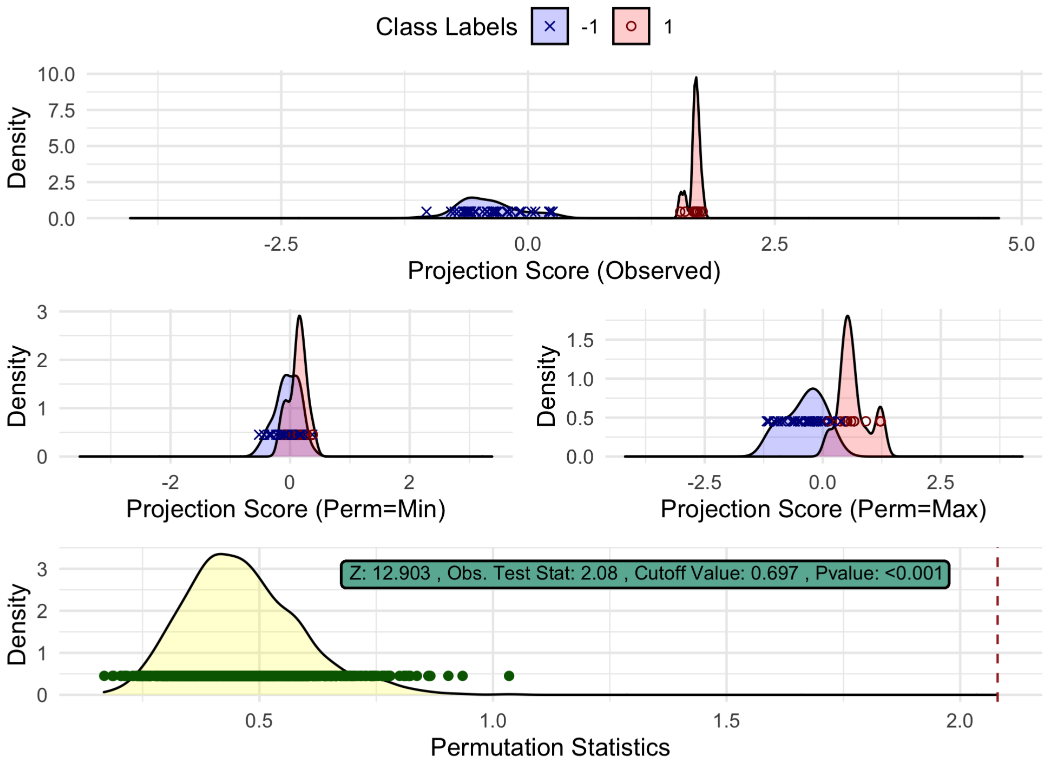


Figure 1: The diagnostic plot from `plotdpp()` for the mushrooms data set. The top graph is the observed projection score distribution of the two classes, the two middle graphs are the projection score distributions of the permutation with the smallest and largest test statistic value, and the bottom graph is the test statistic permutation distribution with the observed statistic value marked by the red dotted line.

Figure 1 displays the default diagnostics for a DiProPerm list. From the observed projection score distribution, one can see clear separation between the two classes. Also, from the projected score distributions of the permutations, which yield the smallest and largest test statistic, we see the score distributions overlap well, so there is some visual justification that the distributions in the observed plot are truly different. Lastly, the bottom plot shows the sampling distribution under the null is located around 0.4 while the observed test statistic is greater than 2. Each individual plot can also be output by the following set of commands:

```
plotdpp(dpp,plots="obs")
plotdpp(dpp,plots="min")
plotdpp(dpp,plots="max")
plotdpp(dpp,plots="permdist")
```

The permutation p-value in Figure 1 suggests that the two high-dimensional distributions of mushroom attributes are indeed different between the two classes. Also displayed is a z-score, calculated by fitting a Gaussian distribution to the test statistic permutation distribution. The mushroom data z-score 12.9 indicates the observed test statistic is approximately 13 standard deviations from the expected value of the test statistic under the null. Finally, the cutoff value 0.697 is displayed, corresponding to the critical value for a hypothesis test at the 0.05 significance level.

Step 4: Examine loadings

In order to assess which variables contributed most toward the separation in step 3, we can print the top five contributors with the code

```
loadings(dpp,loadnum = 5)

    index sorted_loadings
    29    0.5395016
    37    0.3170037
    36    -0.2481763
    111   0.2228389
    20    -0.2087244
```

The top five contributors toward the separation seen in the observed distribution in Figure 1 are indices 29, 37, 36, 111, and 20. These indices correspond to a pungent odor, narrow gill size, broad gill size, urban habitat, and yellow cap color, respectively. For these data, $y = 1$ corresponds to poisonous and $y = -1$ to edible; thus, loadings with positive entries, such as pungent odor, are indicative of poisonous mushrooms. These results are similar to previous analyses, which have also found odor, gill size, habitat, and cap color predictive of mushroom edibility (Pinky et al., 2019; Wibowo et al., 2018).

Summary

DiProPerm is an exact test for comparing two high-dimensional distributions. The **diproperm** package allows the user to visually assess and conduct a DiProPerm test to determine if there is a difference between the high-dimensional distributions of two classes and, if so, evaluate the key features contributing to the separation between the classes.

Acknowledgments

This work was supported by the National Institute of Allergy and Infectious Diseases (NIAID) of the National Institutes of Health (NIH) under award number R37AI054165. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH. The authors thank two anonymous reviewers for helpful comments.

Bibliography

A. Alag. Machine learning approach yields epigenetic biomarkers of food allergy: A novel 13-gene signature to diagnose clinical reactivity. *PLOS ONE*, 14(6):e0218253, 2019. URL https://doi.org/10.1371/journal.pone.0218253. [p266]

A. G. Allmon, J. Marron, and M. G. Hudgens. *diproperm: Conduct Direction-Projection-Permutation Tests and Display Plots*, 2020. URL https://CRAN.R-project.org/package=diproperm. R package version 0.1.0. [p267]

H. An, J. S. Marron, T. A. Schwartz, J. B. Renner, F. Liu, J. A. Lynch, N. E. Lane, J. M. Jordan, and A. E. Nelson. Novel statistical methodology reveals that hip shape is associated with incident radiographic hip osteoarthritis among African American women. *Osteoarthritis and Cartilage*, 24 (4):640–646, 2016. doi: 10.1016/j.joca.2015.11.013. URL https://www.ncbi.nlm.nih.gov/pubmed/26620089https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4799754/. [p267]

M. Aoshima, D. Shen, H. Shen, K. Yata, Y.-H. Zhou, and J. S. Marron. A survey of high dimension low sample size asymptotics. *Australian & New Zealand Journal of Statistics*, 60(1):4–19, 2018. doi: 10.1111/anzs.12212. URL https://doi.org/10.1111/anzs.12212. [p267]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. URL https://CRAN.R-project.org/package=Matrix. R package version 1.2-18. [p269]

P. Bendich, J. S. Marron, E. Miller, A. Pieloch, and S. Skwerer. Persistent Homology Analysis of Brain Artery Trees. *The Annals of Applied Statistics*, 10(1):198–218, 2016. doi: 10.1214/15-AOAS886. URL https://www.ncbi.nlm.nih.gov/pubmed/27642379https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5026243/. [p267]

D. Dua and C. Graff. *UCI Machine Learning Repository*, 2019. URL https://archive.ics.uci.edu/ml/datasets/Mushroom. [p269]

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. New York : Springer, 2001. URL https://catalog.lib.unc.edu/catalog/UNCb4019902. [p267]

A. K. Kimball, L. M. Oko, B. L. Bullock, R. A. Nemenoff, L. F. van Dyk, and E. T. Clambey. A Beginner's Guide to Analyzing and Visualizing Mass Cytometry Data. *The Journal of Immunology*, 200(1):3 LP – 22, 2018. doi: 10.4049/jimmunol.1701494. URL http://www.jimmunol.org/content/200/1/3.abstract. [p266]

X. Y. Lam, J. Marron, D. Sun, and K.-C. Toh. *DWDLargeR: Fast Algorithms for Large Scale Generalized Distance Weighted Discrimination*, 2018a. URL https://CRAN.R-project.org/package=DWDLargeR. R package version 0.1-0. [p268]

X. Y. Lam, J. S. Marron, D. Sun, and K.-C. Toh. Fast Algorithms for Large-Scale Generalized Distance Weighted Discrimination. *Journal of Computational and Graphical Statistics*, 27(2):368–379, 2018b. ISSN 1061-8600. doi: 10.1080/10618600.2017.1366915. URL https://doi.org/10.1080/10618600.2017.1366915. [p268]

E. J. Limkin, R. Sun, L. Dercle, E. I. Zacharaki, C. Robert, S. Reuzé, A. Schernberg, N. Paragios, E. Deutsch, and C. Ferté. Promises and challenges for the implementation of computational medical imaging (radiomics) in oncology. *Annals of Oncology*, 28(6):1191–1206, 2017. doi: 10.1093/annonc/mdx034. URL https://doi.org/10.1093/annonc/mdx034. [p266]

J. S. Marron, M. J. Todd, and J. Ahn. Distance-Weighted Discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007. doi: 10.1198/016214507000001120. URL https://doi.org/10.1198/016214507000001120. [p267]

A. E. Nelson, F. Fang, L. Arbeeva, R. J. Cleveland, T. A. Schwartz, L. F. Callahan, J. S. Marron, and R. F. Loeser. A machine learning approach to knee osteoarthritis phenotyping: data from the FNIH Biomarkers Consortium. *Osteoarthritis and Cartilage*, 27(7):994–1001, 2019. doi: https://doi.org/10.1016/j.joca.2018.12.027. URL http://www.sciencedirect.com/science/article/pii/S1063458419309264. [p267]

N. Pinky, S. M. Islam, and R. Alice. Edibility Detection of Mushroom Using Ensemble Methods. *International Journal of Image, Graphics and Signal Processing*, 11:55–62, 2019. doi: 10.5815/ijigsp.2019.04.05. [p271]

S. Wei, C. Lee, L. Wichers, and J. S. Marron. Direction-Projection-Permutation for High-Dimensional Hypothesis Tests. *Journal of Computational and Graphical Statistics*, 25(2):549–569, 2016. doi: 10.1080/10618600.2015.1027773. URL https://doi.org/10.1080/10618600.2015.1027773. [p266]

A. Wibowo, Y. Rahayu, A. Riyanto, and T. Hidayatulloh. Classification algorithm for edible mushroom identification. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 250–253, 2018. doi: 10.1109/ICOIACT.2018.8350746. [p271]

Andrew G. Allmon
University of North Carolina at Chapel Hill
Department of Biostatistics
dallmon@email.unc.edu

J. S. Marron
University of North Carolina at Chapel Hill
Department of Biostatistics
marron@unc.edu

Michael G. Hudgens
University of North Carolina at Chapel Hill
Department of Biostatistics
mhudgens@email.unc.edu

A Unifying Framework for Parallel and Distributed Processing in R using Futures

by Henrik Bengtsson

Abstract A *future* is a programming construct designed for concurrent and asynchronous evaluation of code, making it particularly useful for parallel processing. The **future** package implements the *Future API* for programming with futures in R. This minimal API provides sufficient constructs for implementing parallel versions of well-established, high-level map-reduce APIs. The future ecosystem supports exception handling, output and condition relaying, parallel random number generation, and automatic identification of globals lowering the threshold to parallelize code. The *Future API* bridges parallel frontends with parallel backends, following the philosophy that end-users are the ones who choose the parallel backend while the developer focuses on what to parallelize. A variety of backends exist, and third-party contributions meeting the specifications, which ensure that the same code works on all backends, are automatically supported. The future framework solves several problems not addressed by other parallel frameworks in R.

Introduction

Parallel processing can be used to speed up computationally intensive tasks. As the size of these tasks and access to more CPU cores tend to grow over time, so does the demand for parallel-processing solutions. In R, there exist several frameworks for running code in parallel, many dating back more than a decade (Schmidberger et al., 2009). R gained built-in support via the **parallel** package in version 2.14.0 (2011), which to date probably provides the most, either directly or indirectly, commonly used solutions. For an overview of current parallel techniques available to R developers, see Eddelbuettel (2021) and the *High-Performance and Parallel Computing with R* CRAN Task View.

The options for parallelizing *computations* in R can be grouped broadly into those that can be used to parallelize R code, such as what the **parallel** package provides, and those that are used to parallelize native code, such as C, C++, and Fortran, and are often not specific to R itself. For example, multi-threaded processing is an efficient parallelization technique which operates at the core of the operating system and the CPU and allows for updating shared memory in parallel and more, which is not available at the R level. In contrast, parallelization at the R level takes place at a higher level with a coarser type of parallelization, which we refer to as *multi-process* parallelization. In addition to parallel computations, there are also efforts in R for working with *parallel data structures*, e.g., **sparklyr** (Luraschi et al., 2021) and the *Programming with Big Data in R* (pbdR) project (Schmidt et al., 2017). By pre-distributing data and storing them on, or near, parallel workers, the overhead from passing data on-the-fly in parallel processing can be decreased, resulting in an overall faster processing time but also lower and more fine-tuned memory requirements. This article proposes a solution for *parallelizing computations at the R level*.

The **future** package (Bengtsson, 2021b) aims to provide a unifying, generic, minimal application protocol interface (API) to facilitate the most common types of parallel processing in R, especially the *manager-worker* strategy where an R process delegates tasks to other R processes. It builds upon the concepts of *futures* (Hewitt and Baker, 1977) and *promises* (Friedman and Wise, 1978; Hibbard, 1976) - concepts that are well suited for a functional language such as R. To better understand how it fits in among and relates to existing parallelization solutions in R[1], let us revisit the two most well-known solutions - packages **parallel** and **foreach**.

The **parallel** package has a set of functions for calling functions and expressions in parallel across one or more concurrent R processes. The most well-known functions for this are `mclapply()` and `parLapply()`, which mimic the behavior of the map-reduce[2] function `lapply()` in the **base** package. Below is an example showing them calling a "slow" function on each element in a vector using two parallel workers. First, to do this through sequentially processing, we can use `lapply()`:

```
xs <- 1:10
y <- lapply(xs, function(x) {
  slow_fcn(x)
})
```

[1]Although the concept of futures could also apply to C, C++, and Fortran parallelization, the future framework targets parallelization at the R level and does not provide an implementation for native code.

[2]We use the term "map-reduce" as it is used in functional programming. The *MapReduce* method by Dean and Ghemawat (2004) was inspired by this term but they are not equivalent.

To do the same in parallel using two *forked* parallel processes, we can use:

```
library(parallel)
xs <- 1:10
y <- mclapply(xs, function(x) {
  slow_fcn(x)
}, mc.cores = 2)
```

Alternatively, to run it in parallel using two R parallel processes running in the *background*, we can do:

```
library(parallel)
workers <- makeCluster(2)
clusterExport(workers, "slow_fcn")
xs <- 1:10
y <- parLapply(workers, xs, function(x) {
  slow_fcn(x)
})
```

These functions, which originate from legacy packages **multicore** (2009-2014, Urbanek (2014)) and **snow** (since 2003, Tierney et al. (2021)), are designed for specific parallelization frameworks. The `mclapply()` set of functions relies on process *forking* by the operating system, which makes them particularly easy to use. This is because each worker automatically inherits the setup and all of the content of the main R process' workspace, making it straightforward to replace a sequential `lapply()` call with a parallel `mclapply()` call. This has made it popular among Linux and macOS developers. On MS Windows, where R does not support forked processing, `mclapply()` falls back to using `lapply()` internally.

The `parLapply()` set of functions, which all operating systems support, rely on a cluster of R background workers for parallelization. It works by the main R process and the workers exchanging tasks and results over a communication channel. The default and most commonly used type of cluster is SOCK, which MS Windows also supports, and it communicates via *socket connections*. Like most other cluster types, SOCK clusters require developers to manually identify and export packages and global objects to the workers by calling `clusterEvalQ()` and `clusterExport()`, before calling `parLapply()`, which increases the barrier to use them.

Mixed responsibilities of developers or end-users

Using either the `mclapply()` or the `parLapply()` approach works well when developers and end-users can agree on which framework to use. Unfortunately, this is not always possible, e.g., R package developers rarely know who the end-users are and what compute resources they have. Regardless, developers who wish to support parallel processing still face the problem of deciding which parallel framework to target, a decision that often has to be done early in the development cycle. This means deciding on what *type of parallelism* to support, e.g., forked processing via `mclapply()` or SOCK clusters via `parLapply()`. This decision is critical because it limits the end-user's options, and any change, later on, might be expensive because of, for instance, having to rewrite and retest part of the codebase. A developer who wishes to support multiple parallel backends has to implement support for each of them individually and provide the end-user with a mechanism to choose between them. This approach often results in unwieldy, hard-to-maintain code of conditional statements with low test coverage, e.g.,

```
if (parallel == "fork") {
  ...
} else if (parallel == "SOCK") {
  ...
} else if (parallel == "MPI") {
  ...
} else {
  ...
}
```

There is no established standard for doing this, which results in different packages providing different mechanisms for controlling the parallelization method, if at all.

Functions like `parLapply()` partly address the problem of supporting multiple parallelization frameworks because they support various types of parallel cluster backends referred to as "snow" clusters (short for *Simple Network of Workstations* and from their origin in the **snow** package), e.g., `workers <- makeCluster(4, type = "FORK")` sets up a cluster that parallelizes using forked processing, and `workers <- makeCluster(4, type = "MPI")` sets up a cluster that parallelizes via a Message Passing Interface (MPI) framework. If a developer uses `parLapply()`, they could write their code such that the end-user can specify what type of snow cluster to use, e.g., by respecting what the end-user set via `setDefaultCluster(workers)`. This provides the end-user with more, although in practice limited, options on how and where to execute code in parallel. Unfortunately, it is rather common that the cluster type is hard-coded inside packages giving end-users little to no control over the parallelization mechanism, other than possibly the number of cores to use.

Map-reduce parallelization with more control for the end-user

Possibly inspired by the snow-style clusters, the **foreach** package (Microsoft and Weston, 2020; Kane et al., 2013), first released in 2009, addresses the above problem of having to decide on the parallel design early on by letting the end-user - not the developer - "register" what type of parallel backend ("foreach adaptor") to use when calling `foreach()`. For example, with **doMC** (Revolution Analytics and Weston, 2020), one can register a multicore cluster, and with **doParallel** (Microsoft Corporation and Weston, 2020), one can register any type of "snow" cluster as in:

```
library(foreach)
library(doParallel)
workers <- parallel::makeCluster(2)
registerDoParallel(workers)

xs <- 1:10
y <- foreach(x = xs) %dopar% {
  slow_fcn(x)
}
```

We note that the specification of what type of parallel framework and number of cores to use is separated from the `foreach()` map-reduce construct itself. This gives more control to the end-user on *how* and *where* to parallelize, leaving the developer to focus on *what* to parallelize, which is a design pattern of great value with important implications on how to design, write, and maintain parallel code. The large uptake of **foreach** since it was first released supports this. As of November 2021, **foreach** is among the top-1.0% most downloaded packages on CRAN, and there are 867 packages on CRAN and Bioconductor that directly depend on it. Another advantage of the separation between the map-reduce frontend API and parallel backend (foreach adaptors) is that new types of parallel backends can be introduced without the need to make updates to the **foreach** package. This has led to third-party developers have contributed additional foreach adaptors, e.g., **doMPI** (Weston, 2017) and **doRedis** (Lewis, 2020).

Unfortunately, there is no *exact* specification on what a foreach adaptor should support and how it should act in certain situations, which has resulted in adaptors behaving slightly differently. At their face value, these differences appear innocent but may cause different outcomes of the same code. In the best case, these differences result in run-time errors, and in the worst case, different results. An example of the former is the difference between **doMC** on Unix-like systems and **doParallel** on Windows. Analogously to `mclapply()`, when using **doMC**, globals and packages are automatically taken care of by the process forking. In contrast, when using **doParallel** with "snow" clusters, globals and packages need to be identified and explicitly exported, via additional arguments `.export` and `.packages` to `foreach()`, to the parallel workers running in the background. Thus, a developer that only uses **doMC** might forget to test their code with **doParallel**, where it may fail. Having said this, the **foreach** package does provide a rudimentary mechanism for automatically identifying and exporting global variables. However, it has some limitations, that, in practice, require the developer to explicitly specify globals to make sure their code works with more backends. Some adaptors provide additional options of their own that are specified as arguments to `foreach()`. If the developer specifies such options, the `foreach()` call might not work with other adaptors.

To develop `foreach()` code invariant to the parallel backend chosen requires a good understanding of how the **foreach** framework works and plenty of testing. This lack of strict behavior is unfortunate and might have grown out of a strategy of wanting to keep things flexible. On the upside, steps have recently[3] been taken toward making the behavior more consistent across foreach backends, suggesting that it is possible to remove several of these weaknesses through a process of deprecating and removing unwanted side effects over several release cycles in close collaboration with package developers currently relying on such backend-specific properties.

[3]See the **foreach** issue tracker at `https://github.com/RevolutionAnalytics/foreach`.

The future framework

The **future** package defines and implements the *Future API* - a minimal, unifying, low-level API for parallel processing, and more. Contrary to the aforementioned solutions, this package does *not* offer a parallel map-reduce API per se. Instead, it focuses on providing efficient and simple-to-use atomic building blocks that allow us to implement such higher-level functions elsewhere.

Three atomic constructs that unify common parallel design patterns

The *Future API* comprises three fundamental constructs:

- `f <- future(expr)` : evaluates an expression via a future (non-blocking, if possible)
- `v <- value(f)` : the value of the future expression `expr` (blocking until resolved)
- `r <- resolved(f)` : TRUE if future is resolved, otherwise FALSE (non-blocking)

To help understand what a future is, let us start with R's assignment construct:

```
v <- expr
```

Although it is effectively a single operator, there are two steps in an assignment: first (i) R evaluates the *expression* on the right-hand side (RHS), and then (ii) it assigns the resulting value to the *variable* on the left-hand side (LHS). We can think of the *Future API* as giving us full access to these two steps by rewriting the assignment construct as:

```
f <- future(expr)
v <- value(f)
```

Contrary to the regular assignment construct where the evaluation of the expression and the assignment of its value are tightly coupled, the future construct allows us to decouple these steps, which is an essential property of futures and necessary when doing parallel processing[4]. Especially, the decoupling allows us to perform other tasks in-between the step that evaluates the expression and the step that assigns its value to the target variable. Here is an example that creates a future that calculates `slow_fcn(x)` with `x` being `1`, then reassigns a different value to `x`, and finally gets the value of the future expression:

```
x <- 1
f <- future({
  slow_fcn(x)
})
x <- 2
v <- value(f)
```

By definition, a future consists of an R expression and any required objects as they were when the future was created. Above, the recorded objects are the function `slow_fcn()` and the variable `x` with value `1`. This is why the value of the future is unaffected by `x` getting reassigned a new value after the future is created but before the value is collected.

We have yet to explain how futures are resolved, that is, how the future expression is evaluated. This is the part where futures naturally extend themselves to asynchronous and parallel processing. How a future is resolved depends on what *future backend* is set. If not specified, the default is to resolve futures sequentially, which corresponds to setting:

```
plan(sequential)
```

Before we continue, it should be emphasized that the *Future API* is designed so that a program using it gives the same results no matter how and where the futures are resolved, may it be sequentially on the local machine or in parallel on a remote cluster. As a consequence, *the future ecosystem is designed to separate the responsibilities of the developer from those of the end-user*. This allows the developer to focus on the code to be parallelized while the end-user focuses on how to parallelize. It is the end-user who decides on the `plan()`. For example, if they specify:

```
plan(multisession)
```

[4]We can find this future-value pattern in several implementations for parallel processing, including the ones we use in R. The `mcparallel()`-`mccollect()` pair of functions in **parallel** is one example. This is why the future-value abstraction can be mapped onto many of our existing parallel frameworks in a unified way.

before calling the above future code, futures will be resolved in parallel via a SOCK cluster on the local machine similar to what we used above in the `parLapply()` example. If the end-user instead specifies `plan(multicore)`, futures will be resolved in parallel in the background via *forked* R processes using the same framework as `mclapply()`. Importantly, regardless of what future plan is used, and regardless of whether or not we assigned a new value to `x` after creating the future, the result is always the same. Since we, as developers, do not know what backend end-users will use, we also cannot know *when* a future is resolved. This is why we say that "a future evaluates its expression *at some point in the future*". What we do know is that `value()` returns the value of the future only when it is resolved, and if it is not resolved, then `value()` waits until it is.

Next, let us look at how blocking works by using an example where we create three futures to be resolved by two parallel workers:

```
library(future)
plan(multisession, workers = 2)

xs <- 1:10

f1 <- future({
  slow_fcn(xs[1])
})

f2 <- future({
  slow_fcn(xs[2])
})

f3 <- future({
  slow_fcn(xs[3])
})
```

Here, the first two futures are created in a non-blocking way because there are two workers available to resolve them. However, when we attempt to create a third future, there are no more workers available. This causes `future()` to block until one of the workers is available, that is, until either one or both of the two futures have been resolved. If three or more workers are set up, then the third `future()` call would not block. On the other hand, if `plan(sequential)` is set, then each `future()` blocks until the previously created future has been resolved. Finally, to retrieve the values of the three futures, we do:

```
v1 <- value(f1)
v2 <- value(f2)
v3 <- value(f3)
```

Although it is common to call `value()` on the futures in the order we created them, we can collect the values in any order, which is something we will return to later.

Continuing, we can generalize the above to calculate `slow_fcn()` on each of the elements in `xs` via futures. For this, we can use a regular for-loop to create each of the `length(xs)` futures:

```
xs <- 1:10
fs <- list()
for (i in seq_along(xs)) {
  fs[[i]] <- future(slow_fcn(xs[i]))
}
```

Note how we here have effectively created a *parallel for-loop*, where `plan()` controls the amount of parallelization. To collect the values of these futures, we can use[5]:

```
vs <- lapply(fs, value)
```

Alternatively, to using a for-loop, we can parallelize using `lapply()`:

```
xs <- 1:10
fs <- lapply(xs, function(x) {
  future(slow_fcn(x))
})
```

This is illustrated in Figure 1, where four background workers created by `plan(multisession,workers = 4)` is used to resolve the futures. The same idea also applies to other types of map-reduce functions.

[5]Here, `vs <- lapply(fs, value)` is used for clarification but we could also have used `vs <- value(fs)` because `value()` is a generic function with implementation also for lists and other types of containers.

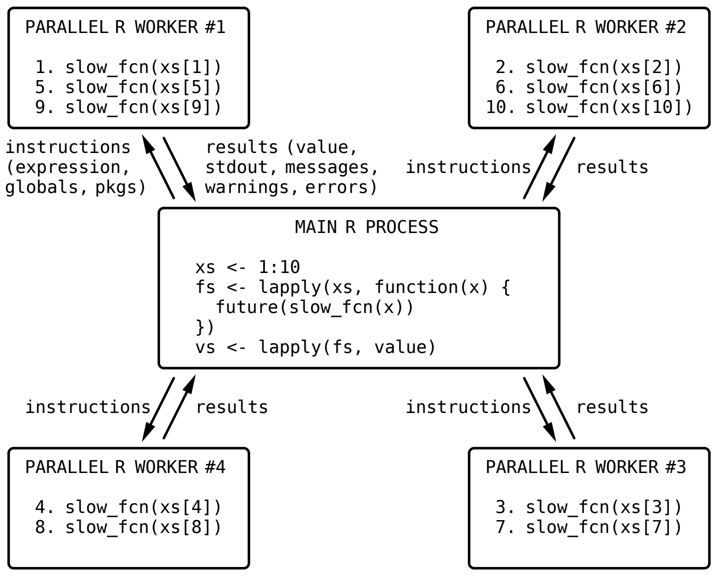


Figure 1: An illustration of parallel processing using futures via four R processes running in the background. Base R `lapply()` is used to call `slow_fcn()` ten times - once per element in `xs`. By calling it via `future()`, each call is distributed out to one of four workers. If all workers are busy, the next future, in turn, will wait for a worker to become available. The results of all futures are collected at the end. Any output, warnings, and errors produced on the workers are relayed as-is back on the main R session. The four workers were created using `plan(multisession, workers = 4)`. If switching to `plan(sequential)`, then all futures are resolved sequentially in the main R process. Only core Future API functions from the **future** package were used. Less verbose, map-reduce alternatives are available in the high-level future packages such as **future.apply**, **furrr**, and **doFuture**.

This shows how powerful the *Future API* is; by combining base R with the two constructs `future()` and `value()`, we have created rudimentary[6] alternatives to `mclapply()`, `parLapply()`, and `foreach()`. Indeed, we could reimplemented these **parallel** and **foreach** functions using the *Future API*.

The `resolved()` function queries, in a non-blocking way, whether or not a future is resolved. Among other things, this can be used to collect the value of a subset of resolved futures as soon as possible without risking to block from collecting the value of a non-resolved future, which allows additional futures to launch sooner, if they exist. This strategy also helps lower the overall latency that comes from the overhead of collecting values from futures - values that may contain large objects and are collected from remote machines over a network with limited bandwidth. As explained further below, collecting the value of futures as soon as possible will also lower the latency of the relay of output and conditions (e.g., warnings and errors) captured by each future while they evaluate the future expressions.

In summary, the three constructs of the *Future API* provide *the necessary and sufficient* functionality for evaluating R expressions in parallel, which in turn may be used to construct higher-level map-reduce functions for parallel processing. Additional core features of futures that are useful, or even essential, for parallel processing are presented next.

Exception handling

To make it as simple as possible to use futures, they are designed to mimic the behavior of the corresponding code that does not use futures. An important part of this design aim is how exception handling is done. Any *error* produced while resolving a future, that is, evaluating its expression, is captured and relayed as-is in the main R process each time `value()` is called. This mimics the behavior of how errors are produced when not using futures. This is illustrated by the following two code examples – with futures:

[6]These solutions process each element in a separate future, which is suboptimal if the overhead of creating a future is relatively large compared to the evaluation time. This overhead can be mitigated by processing elements in chunks, something that requires more complex code than what is shown in these examples.

```
x <- "24"
f <- future(log(x))
v <- value(f)
# Error in log(x) : non-numeric argument to mathematical function
```

and without futures:

```
x <- "24"
v <- log(x)
# Error in log(x) : non-numeric argument to mathematical function
```

As a result, standard mechanisms for condition handling also apply to errors relayed by futures. For example, to assign a missing value to v whenever there is an error, we can use:

```
v <- tryCatch({
  value(f)
}, error = function(e) {
  NA_real_
})
```

Errors due to extraordinary circumstances, such as terminated R workers and failed communication, are of a different kind than the above evaluation errors. Because of this, they are signaled as errors of class *FutureError* so they can be handled specifically, e.g., by restarting R workers or relaunching the failed future elsewhere (Section 'Future work').

Relaying of standard output and conditions (e.g., messages and warnings)

Futures capture the standard output (*stdout*) and then relay it in the main R process each time `value()` is called. Analogously, all conditions are captured and relayed as-is in the main R process each time `value()` is called. Common conditions relayed this way are *message*s and *warning*s as generated by `message()` and `warning()`. The relaying of errors was discussed in the previous section. Relaying of standard output and conditions respects the order they were captured, except that all of the standard output is relayed before conditions are relayed in the order they were signaled. For example,

```
x <- c(1:10, NA)
f <- future({
  cat("Hello world\n")
  y <- sum(x, na.rm = TRUE)
  message("The sum of 'x' is ", y)
  if (anyNA(x)) warning("Missing values were omitted", call. = FALSE)
  cat("Bye bye\n")
  y
})
v <- value(f)
# Hello world
# Bye bye
# The sum of 'x' is 55
# Warning message:
# Missing values were omitted
```

Standard techniques can be used to capture the relayed standard output, e.g.,

```
stdout <- capture.output({
  v <- value(f)
})
# The sum of 'x' is 55
# Warning message:
# Missing values were omitted

stdout
# [1] "Hello world" "Bye bye"
```

Similarly, `withCallingHandlers()` and `globalCallingHandlers()` can be used to capture and handle the different classes of conditions being relayed. Note that all of the above works the same way regardless of what future backend is used, including when futures are resolved on a remote machine.

Relaying of standard output, messages, warnings, and errors simplifies any troubleshooting. For example, existing verbose output helps narrow down the location of errors and warnings, which may reveal unexpected missing values or vector recycling. Commonly used poor-man debugging, where temporary debug messages are injected into the code, is also possible because of this built-in relay mechanism. Imagine a logging framework that leverages R's condition framework to signal different levels of log events and then captures and reports, e.g., to the terminal or to file. It will work out of the box when parallelizing with futures.

Conditions of class *immediateCondition* are treated specially by the future framework. They are by design allowed to be relayed as soon as possible, and not only when `value()` is called. For instance, they may be relayed when calling `resolved()`, or even sooner, depending on the future backend used. Because of this, *immediateCondition* conditions are relayed without respecting the order of other types of conditions captured. This makes them suitable for signaling, for instance, *progress updates*. Thus, such progress conditions can be used to update a progress bar in the terminal or in a Shiny application while originating from futures being resolve on remote machines. See the **progressr** package (Bengtsson, 2021h) for an implementation of this. Note, however, that this type of near-live relaying of *immediateCondition*s only works for backends that have the means to communicate these conditions from the worker back to the main R session, while the worker still processes the future. When non-supporting backends are used, these conditions are relayed together with other captured conditions at the very end when the future has been resolved.

Comment: Contrary to the standard output, due to limitations in R[7], it is not possible to capture the standard error reliably. Because of this, any output to the standard error is silently ignored, e.g., `cat("some output", file = stderr())`. However, although output from `message()` is sent to the standard error, it is indeed outputted in the main R processes because it is the message conditions that are captured and relayed, not the standard error.

Globals and packages

The future framework is designed to make it as simple as possible to implement parallel code. Another example of this is the automatic identification of *globals* - short for global variables and functions - that are required for a future expression to be resolved successfully. For example, in:

```
f <- future({
  slow_fcn(x)
})
```

the globals of the future expression are `slow_fcn()` and `x`. By default, `future()` will attempt to identify, locate, and record these globals internally via static code inspection, such that they are available when the future is resolved. If one of these globals is part of a package namespace, that is also recorded. Because of this, developers rarely need to worry about globals when programming with futures. However, occasionally, the future expression is such that it is not possible to infer all the globals. For example, the following produces an error:

```
plan(multisession)
k <- 42
f <- future({
  get("k")
})
v <- value(f)
# Error in get("k") : object 'k' not found
```

This is because code inspection cannot infer that `k` is a needed variable. In such cases, one can guide the future framework to identify this missing global by explicitly mentioning it at the top of the future expression, e.g.,

```
f <- future({
  k
  get("k")
})
```

Alternatively, one can specify it via argument `globals` when creating the future, e.g.,

```
f <- future({
  get("k")
}, globals = "k")
```

See `help("future", package = "future")` for all options available to control which globals to use and how to ignore false positives.

[7]See https://github.com/HenrikBengtsson/Wishlist-for-R/issues/55

Internally, the future framework uses **globals** (Bengtsson, 2020), and indirectly **codetools** (Tierney, 2020), to identify globals by walking the abstract syntax tree (AST) of the future expression in order. It uses an *optimistic* search strategy to allow for some false-positive globals to minimize the number of false-negative globals. Contrary to false positives, false negatives cause futures to produce errors similar to the one above.

Proper parallel random number generation

The ability to produce high-quality random numbers is essential for the validity of many statistical analyses, e.g., bootstrap, permutation tests, and simulation studies. R has functions at its core for drawing random numbers from common distributions. This R functionality is also available to C and Fortran native code. All draw from the same internal pseudo-random number generator (RNG). Different kinds of RNGs are available, with Mersenne-Twister (Matsumoto and Nishimura, 1998) being the default. Like most other RNGs, the Mersenne-Twister RNG is not designed for concurrent processing - if used in parallel, one risks producing random numbers that are correlated. Instead, for parallel processing, the multiple-recursive generator L'Ecuyer-CMRG by L'Ecuyer (1999), implemented in the **parallel** package, can be used to set up multiple RNG streams. The future ecosystem has built-in support for L'Ecuyer-CMRG at its core to make it as easy as possible to produce statistically sound and reproducible random numbers regardless of how and where futures are resolved, e.g.,

```
f <- future(rnorm(3), seed = TRUE)
value(f)
#  [1] -0.02648871 -1.73240257  0.78139056
```

Above, `seed = TRUE` is used to specify that parallel RNG streams should be used. When used, the result will be fully reproducible regardless of future backend specified and the number of workers available. Because `seed = TRUE` can introduce significant overhead, the default is `seed = FALSE`. However, since it is computationally cheap to detect when a future expression produced random numbers, the future framework will generate an informative warning when this is used by mistake to help lower the risk of producing statistically questionable results. It is possible to disable this check or to escalate the warning to an error via an R option. All higher-level parallelization APIs that build upon futures must adhere to this parallel-RNG design, e.g., **future.apply** and **furrr**.

Future assignment construct

As an alternative for using `future()` and `value()`, the **future** package provides a *future-assignment operator*, `%<-%`, for convenience. It is designed to mimic the regular assignment operator, `<-`, in R:

```
v <- expr
```

By replacing the above with:

```
v %<-% expr
```

the RHS expression `expr` will be evaluated using a future whose value is assigned to the LHS variable `v` as a *promise*[8]. Because the LHS is a promise, the value of the future will not be assigned to it until we attempt to access the promise. As soon as we try to use `v`, say,

```
y <- sqrt(v)
```

the associated promise will call `value()` on the underlying future, while possibly blocking, and at the end assign the collected result to `v`[9]. From there on, `v` is a regular value. As an illustration, our introductory example with three futures can be written as[10]:

```
xs <- 1:10
v1 %<-% slow_fcn(xs[1])
v2 %<-% slow_fcn(xs[2])
v3 %<-% slow_fcn(xs[3])
```

and with, say, `plan(multisession)`, these statements will be processed in parallel.

Special *infix operators* are available to specify arguments that otherwise would be passed to the `future()` function. For example, to set `seed = TRUE`, we can use:

[8]The type of promises that R supports should not be mistaken for the type of promises as defined by the **promises** (Cheng, 2021) package, which, together with futures, is used for asynchronous processing in Shiny applications.

[9]The internal call to `value()` will also cause any captured standard output and conditions to be relayed.

[10]I have dropped the curly brackets on the RHS to make the example tidier. Just like with regular assignment, there is nothing preventing us from using composite expressions also with future assignments.

```
v %<-% rnorm(3) %seed% TRUE
```

See `help("%<-%", package = "future")` for other infix operators.

Regular R assignments can often be replaced by future assignments as-is. However, because future assignments rely on promises, and promises can only be assigned to *environment*s, including the working environment, they cannot be used to assign to, for instance, *list*s. As a workaround, one can use a *list environment* instead of a *list*. They are implemented in the **listenv** package (Bengtsson, 2019). A list environment is technically an *environment* that emulates most properties of a *list*, including indexing as in:

```
xs <- 1:10
vs <- listenv::listenv()
for (i in seq_along(xs)) {
  vs[[i]] %<-% slow_fcn(xs[i])
}
vs <- as.list(vs)
```

Nested parallelism and protection against it

A problem with parallel processing in software stacks like the R package hierarchy is the risk of overloading the CPU cores due to nested parallelism. For instance, assume that package **PkgA** calls `PkgB::estimate()` in parallel using all N cores on the current machine. Initially, the `estimate()` function was implemented to run sequentially, but, in a recent **PkgB** release, it was updated to parallelize internally using all N cores. Without built-in protection, this update now risks running N^2 parallel workers when **PkgA** is used, possibly without the awareness of either maintainer.

The **future** package has built-in protection against nested parallelism. This works by configuring each worker to run in sequential mode unless nested parallelism is explicitly configured. This is achieved by setting options and environment variables that are known to control parallelism in R, e.g., `options(mc.cores = 1)`. Because of this, if **PkgA** and **PkgB** parallelize using the future framework, the nested parallelism above will run with a total of N cores, not N^2 cores. This will also be true for non-future code that respects such settings, e.g., when **PkgB** uses `parallel::mclapply()` with the default `mc.cores` argument.

Nested parallelism can be configured by the end-user via `plan()`. For example, to use two workers for the first layer of parallelization and three for the second, use:

```
plan(list(
  tweak(multisession, workers = 2),
  tweak(multisession, workers = 3)
))
```

This will run at most $2 \times 3 = 6$ tasks in parallel on the local machine. Any nested parallelism beyond these two layers will be processed in sequential mode. That is, `plan(sequential)` is implicit if not specified. When argument `workers` is not specified, it defaults to `parallelly::availableCores()`, which respect a large number of environment variables and R options specifying the number of cores. Because of this, and due to the built-in protection against nested parallelism, using `plan(list(multisession, multisession))` effectively equals using `plan(list(multisession, sequential))`.

A more common scenario of nested parallelism is when we submit tasks to a job scheduler on a compute cluster where each job is allowed to run on multiple cores allotted by the scheduler. As clarified later, this may be configured as:

```
plan(list(
  future.batchtools::batchtools_sge,
  multisession
))
```

where the default `workers = availableCores()` assures that the number of multisession workers used respects what the scheduler assigns to each job.

Future backends

In addition to implementing the *Future API*, the **future** package also implements a set of future backends that are based on the **parallel** package. If no backend is specified, the default is:

```
plan(sequential)
```

which makes all futures to be resolved sequentially in the current R session. To resolve futures in parallel on a SOCK cluster on the local machine, use one of:

```
plan(multisession) ## defaults to workers = availableCores()
plan(multisession, workers = 4)
```

Similarly, to resolving futures in parallel on the local machine via *forked* processing, use one of:

```
plan(multicore) ## defaults to workers = availableCores()
plan(multicore, workers = 4)
```

To resolve futures via *any* type of "snow" cluster, use the `cluster` backend. For example, to use a traditional SOCK cluster or an MPI cluster, use either of:

```
workers <- parallel::makeCluster(4)
plan(cluster, workers = workers)

workers <- parallel::makeCluster(4, type = "MPI")
plan(cluster, workers = workers)
```

To use a SOCK cluster with two remote workers, use:

```
plan(cluster, workers = c("n1.remote.org", "n2.remote.org"))
```

which is short for:

```
workers <- parallelly::makeClusterPSOCK(c("n1.remote.org", "n2.remote.org"))
plan(cluster, workers = workers)
```

This works as long as there is password-less SSH access to these remote machines and they have R installed. Contrary to `parallel::makePSOCKcluster()`, `parallelly::makeClusterPSOCK()` uses reverse-tunneling techniques, which avoids having to configure inward-facing port-forwarding in firewalls, something that requires administrative rights.

Third-party future backends

Besides these built-in future backends, other R packages available on CRAN implement additional backends. As long as these backends conform to the *Future API* specifications, as discussed in Section 'Validation', they can be used as alternatives to the built-in backends. For example, the **future.callr** package (Bengtsson, 2021e) implements a future backend that resolves futures in parallel on the local machine via R processes[11], orchestrated by the **callr** (Csárdi and Chang, 2021) package, e.g.,

```
plan(future.callr::callr)  ## defaults to workers = availableCores()
plan(future.callr::callr, workers = 4)
```

Another example is **future.batchtools** (Bengtsson, 2021d), which implements several types of backends on top of the **batchtools** (Lang et al., 2017) package. Most notably, it provides backends that resolve futures distributed on high-performance compute (HPC) environments by submitting the futures as jobs to a job scheduler, e.g., Slurm, SGE, and Torque/PBS:

```
plan(future.batchtools::batchtools_slurm)
plan(future.batchtools::batchtools_sge)
plan(future.batchtools::batchtools_torque)
```

Yet another example is the **googleComputeEngineR** package (Edmondson, 2019), which provides a "snow" cluster type that supports[12] resolving futures in the cloud on the Google Compute Engine platform.

[11]The callr backend performs similarly to the PSOCK-based multisession backend. However, in contrast to the latter, it does not rely on socket connections, which on MS Windows may require administrative rights on the machine's firewall in order to allow the R process to communicate on certain ports. Moreover, on machines with a large number of cores, PSOCK clusters are limited to 125 parallel workers because that is the maximum number of connections R can have open simultaneously.

[12]It also supports using `parLapply()` functions.

Implementation

The future framework is platform-independent and works on all platforms, including Linux, Solaris, macOS, and MS Windows. It is backward compatible with older versions of R back to R 3.1.2 (October 2014). The core packages **future**, **parallelly** (Bengtsson, 2021g), **globals**, and **listenv** are implemented in plain R (without native code) to maximize cross-platform operability and to keep installation simple. They are available on CRAN (since 2015). The **parallelly** package implements enhancements to the **parallel** package originally part of the **future** package. The **digest** package (Eddelbuettel et al., 2021) is used to produce universally unique identifiers (UUIDs). Development is done toward a public Git repository hosted at `https://github.com/HenrikBengtsson/future`.

Validation

Since correctness and reproducibility is essential to all data processing, validation is a top priority and part of the design and implementation throughout the future ecosystem. Several types of testing are performed.

First, all the essential core packages part of the future framework, **future**, **parallelly**, **globals**, and **listenv**, implement a rich set of package tests. These are validated regularly across the wide range of operating systems (Linux, Solaris, macOS, and MS Windows) and R versions available on CRAN, on continuous integration (CI) services (GitHub Actions, Travis CI, and AppVeyor CI), and on R-hub.

Second, for each new release, these packages undergo full reverse-package dependency checks using **revdepcheck** (Csárdi and Wickham, 2021). As of November 2021, the **future** package is tested against 210 direct reverse-package dependencies available on CRAN and Bioconductor. These checks are performed on Linux with both the default settings and when forcing tests to use multisession workers (SOCK clusters), which further validates that globals and packages are identified correctly.

Third, a suite of *Future API conformance tests* available in the **future.tests** package (Bengtsson, 2021f) validates the correctness of all future backends. Any new future backend developed must pass these tests on complying with the *Future API*. By conforming to this API, the end-user can trust that the backend will produce the same correct and reproducible results as any other backend, including the ones that the developer has tested on. Also, by making it the responsibility of the backend developer to assert that their new future backend conforms to the *Future API*, we relieve other developers from having to test that their future-based software works on all backends. It would be a daunting task for a developer to validate the correctness of their software with all existing backends. Even if they would achieve that, there may be additional third-party future backends that they are not aware of, that they do not have the possibility to test with, or that yet have not been developed.

Fourth, since **foreach** is used by a large number of essential CRAN packages, it provides an excellent opportunity for supplementary validation. Specifically, we dynamically tweak the examples of **foreach** and popular CRAN packages **caret**, **glmnet**, **NMF**, **plyr**, and **TSP** to use the **doFuture** adaptor (Bengtsson, 2021a). This allows us to run these examples with a variety of future backends to validate that the examples produce no run-time errors, which indirectly validates the backends as well as the *Future API*. In the past, these types of tests helped to identify and resolve corner cases where automatic identification of global variables would fail. As a side note, several of these foreach-based examples fail when using a parallel foreach adaptor because they do not properly export globals or declare package dependencies. The exception is when using the sequential *doSEQ* adaptor (default), fork-based ones such as **doMC**, or the generic **doFuture**, which supports any future backend and relies on the future framework for handling globals and packages[13].

Lastly, analogously to the above reverse-dependency checks of each new release, CRAN and Bioconductor continuously run checks on all these direct, but also indirect, reverse dependencies, which further increases the validation of the *Future API* and the future ecosystem at large.

Known limitations

When saving an R object to file or sending it to a parallel worker, R uses a built-in technique called *serialization*, which allows a complex object structure to be sent as a stream of bytes to its destination, so it later can be reconstructed via *unserialization*. The ability to serialize objects is fundamental to all parallel processing, the exception being shared-memory strategies such as forked parallel processing. For example, this is how future expressions and variables are sent to parallel workers and how results are returned.

[13]There is a plan to update **foreach** to use the exact same static-code-analysis method as the **future** package use for identifying globals. As the maintainer of the future framework, I collaborate with the maintainer of the **foreach** package to implement this.

However, some types of objects are by design bound to the R session where they are created and cannot be used as-is in other R processes. One example is R *connections*, e.g.,

```
con <- file("/path/to/file", open = "wb")
str(con)
#  'file' int 3
#  - attr(*, "conn_id")=<externalptr>
```

Any attempt to use a connection in another R process, for instance, by saving it to file, restarting R, and loading it back in, or by sending it to a parallel worker, will at best produce a run-time error, and in the worst case, produce invalid results or, for instance, write to the wrong file. These constraints apply to all types of parallelization frameworks in R, including the future framework.

There are other types of objects that cannot be transferred as-is to external processes, many from popular third-party packages, e.g., database connections of the **DBI** package, XML documents of the **xml2** package, STAN models of the **stan** package, and many more[14]. An indicator of this is when an R object has an *external pointer*, which is used for referencing an internal low-level object. This suggests that the object is bound to the current process and its lifespan. Unfortunately, it is not a sufficient indicator because some objects with external pointers can be exported, e.g., **data.table** objects. This makes it complicated to automate the detection of non-exportable objects and protect against using them in parallel processing. The current best practice is to be aware of these types of objects and to document new ones when discovered, which often happens when there is an unexpected run-time error. To help troubleshooting, it is possible to configure the **future** package to scan for and warn about globals with external pointers whenever used in a future.

Finally, it is theoretically possible to restructure some of the "non-exportable" object types such that they can be used in parallel processing. This is discussed further in the 'Future work' section.

Overhead

With parallel processing comes overhead. Typically, sources of added processing time are from spawning new parallel processes, sending instructions and globals to the workers, querying workers for results, and receiving results (Figure 1). Because of this, there is always a trade-off between sequential and parallel processing, and on how many parallel workers can be used before the total overhead dominates the benefits. Whether or not parallelization is beneficial, and for which parallel backends, depends on what is being parallelized.

As with other parallel solutions, in the future framework, overhead differs between parallel backends. Certain parallel backends, such as forked processing ("multicore"), are better suited for low-latency requirements, whereas others, such as distributed processing ("cluster" and "batchtools"), are better suited for large-throughput requirements. For example, many fast operations applied to a single large data frame should probably be parallelized on the local computer with forked processing, if supported, rather than being distributed on a compute cluster running in the cloud. In contrast, processing hundreds of data files may be completed sooner if distributed out to multiple computers (with access to the same file system), for instance, via a job scheduler, rather than being processed in parallel on the local machine.

Besides the overhead added by the parallel backend, each future, regardless of backend, has a baseline overhead. Specifically, there is a small overhead from the static-code inspection used to identify global variables, from exception handling needed to capture and relay errors, and from capturing and relaying standard output and conditions. Except for the error-handling overhead, these can all be avoided via certain `future()` arguments, e.g., by manually specifying globals needed and by disabling the relaying of output and conditions.

R has several profiling tools that can help identify bottlenecks and overhead in computational expensive tasks, e.g., `system.time()` of the **base** package, **microbenchmark** (Mersmann, 2021), **bench** (Hester, 2020), `Rprof()` of the **utils** package, **proffer** (Landau, 2021a), and **profvis** (Chang et al., 2020). These tools can also identify the different sources of overhead in the parallelization framework itself, including the ones in the future ecosystem. It is on the roadmap to make futures collect and report on some of these benchmarks automatically in order to help developers optimize their code and for end-users to choose a proper backend.

[14]See **future** package vignette '`Non-exportable object`' for more examples.

Results

The *Future API* is designed to unify parallel processing in R at the lowest possible level. It provides a standard for building richer, higher-level parallel frontends without having to worry about and reimplement common, critical tasks such as identifying global variables and packages, parallel RNG, and relaying of output and conditions - cumbersome tasks that are often essential to parallel processing.

Another advantage of the future framework is that new future backends do not have to implement their versions of these tasks, which not only lowers the threshold for implementing new backends, but also results in a consistent behavior throughout the future ecosystem, something none of the other parallel solutions provide. This benefits the developer because they can focus on what to parallelize rather than how and where. It also benefits the end-user, who will have more alternatives to how and where parallelization will take place. For instance, the developer might have local parallelization in mind during the development phase due to their work-environment constraints, whereas the end-user might be interested in parallelizing out to a cloud computing service. One may say that code using futures scales far without the developer's attention. Moreover, code using futures for parallelization will be able to take advantage of new backends that may be developed several years from now.

Directly related to the separation of code and backends, end-users and developers no longer need to rely on other package maintainers to update their code to take advantage of any new types of computational resources; updates that otherwise require adding another argument and conditional statement. One example of this was **future.batchtools**' predecessor, **future.BatchJobs** (legacy, CRAN, archived), which was straightforward to implement on top of **BatchJobs** (Bischl et al., 2015) as soon as the *Future API* was available. With zero modifications, code that previously only parallelized on the local computer could suddenly parallelize across thousands of cores on high-performance compute (HPC) clusters via the job scheduler. All it took was to change the `plan()`.

Because the future ecosystem is at its core designed to give consistent results across all sequential and parallel backends, it is straightforward to update, or port, an existing, sequential, map-reduce framework such that it can run in parallel. Not having to worry about low-level parallelization code, which otherwise risks blurring the objectives, lowers the threshold for designing and implementing new parallel map-reduce APIs. There are several examples of how fairly straightforward it is to implement higher-level parallel APIs on top of the *Future API*. The **future.apply** package (Bengtsson, 2021c), implements futurized variants of R's apply functions found in the **base** package, e.g., `future_apply()` and `future_lapply()` are plug-in replacements for `apply()` and `lapply()`. The **furrr** package (Vaughan and Dancho, 2021) implements futurized variants of the different map-reduce functions found in the **purrr** package (Henry and Wickham, 2020), e.g., `future_map()` is as plug-in replacement for `map()`. The **doFuture** package implements a generic **foreach** adaptor for `y <- foreach(...) %dopar% { ... }` that we can use with any future backend. Because the **BiocParallel** (Morgan et al., 2021) package, part of the Bioconductor Project, supports foreach as its backend, its functions such as `bplapply()` and `bpvec()` can also parallelize using *any type of future backend* via **doFuture**.

By lowering the barrier for implementing futurized variants of popular map-reduce APIs, developers and end-users are allowed to stay with their favorite coding style while still taking full advantage of the future framework.

The *Future API* also addresses the lock-in-versus-portability problem mentioned in the introduction; the risk that package developers on Unix-like systems would only support multicore parallelization methods because "`mclapply()` just works" is significantly lower using futures. Similarly, the most common way to parallelize code is to use multiple cores on the local machine. Because it is less common to have access to multiple machines, this often prevents developers from considering any other types of parallelization, with the risk of locking in end-users with other types of resources to only use a single machine. Hence, the chance for a package to support multi-host parallelization, including in the cloud and HPC environments, increases when using futures.

The burden on package developers to test and validate their parallel code is significant when using traditional parallelization frameworks, especially when attempting to support multiple variants. In contrast, when using futures, the cost of developing, testing, and maintaining parallel code is lower - often not much more than maintaining sequential code. This is possible because of the simplicity of the *Future API* and the fact that the orchestration of futures is predominantly done by the **future** package. Therefore, by implementing rigorous tests for the future framework and the different backend packages, the need for performing complementary tests in packages that make use of futures is much smaller. Tests for future backend packages, as well as the *Future API*, are provided by the **future.tests** package, which lowers the risk for a backend not being sufficiently tested.

The built-in protection against nested parallelism by mistake, and the agility of system settings of `availableCores()`, makes parallel code that uses futures to play nicely on multi-tenant systems. It respects all known R options and environment variables that specify, or otherwise limit the number of parallel workers allowed. See `help("availableCores", package = "parallelly")` for details. In contrast, it is, unfortunately, very common to find parallel code that uses `parallel::detectCores()` as the default number of workers in other parallel frameworks. Defaulting to using all available cores this way often wreak havoc on multi-tenant compute systems by overusing already consumed CPU resources, sometimes bringing the system to a halt due to too much context switching and memory use. Unfortunately, this often results in a negative performance on also other users' processes, and system administrators have to spend time tracking down the root cause of such poorly performing compute hosts.

Use of the future framework on CRAN and Bioconductor

The **future** package was released on CRAN in 2015. The uptake has grown steadily ever since. As of November 2021, **future** is among the top-1.1% most downloaded package on CRAN[15], and there are 210 packages on CRAN and Bioconductor that directly depend on it. For map-reduce parallelization packages **future.apply** (top-1.3% most downloaded) and **furrr** (top-1.8%), the corresponding number of packages are 87 and 58, respectively.

Besides supporting these traditional parallelization methods, the future framework is also used as an infrastructure elsewhere. For example, the workflow package **targets** (Landau, 2021b), and its predecessor **drake** (Landau, 2018), implements "a pipeline toolkit for reproducible computation at scale". They work by defining make-like targets and dependencies that can be resolved in parallel using any type of future backend. Another prominent example is the **shiny** package (Chang et al., 2021), which implements support for *asynchronous processing* in Shiny applications via futures. Asynchronous processing helps to avoid long-running tasks from blocking the user interface. Similarly, the **plumber** package (Schloerke and Allen, 2021), which automatically generates and serves HTTP API from R functions, uses futures to serve asynchronous web APIs and process tasks in parallel.

Other uses of futures

In Hewitt and Baker (1977), the authors propose the `(EITHER ...)` construct that "evaluates the expressions in parallel and return the value of 'the first one that finishes'." A corresponding R construct could be `future_either(...)` that evaluates R expressions concurrently via futures and returns the value of the first resolved one ignoring the others, e.g.,

```
y <- future_either(
  sort.int(x, method = "shell"),
  sort.int(x, method = "quick"),
  sort.int(x, method = "radix")
)
```

We may also use futures in cases that do not require parallel processing per se. Indeed, the *Future API* strives to make no assumptions about futures being resolved via parallel or distributed processing. One example is where a particular expression can only be resolved in a legacy version of R, on another operating system than where the main R session runs, or in an environment that meet specific requirements, e.g., large amounts of memory, fast local disks, or access to a certain genomic database. Another example of a resource-specific backend is the **civis** package (Miller and Ingersoll, 2020), which uses futures to provide an R client for the commercial Civis Platform.

We can also use futures to evaluate non-trustworthy R expressions in a sandboxed R environment that is, for instance, locked down in a virtual machine, or in a Linux container, such as Singularity (Kurtzer et al., 2017) or Docker (Merkel, 2014), without access to the host machine and its file system and network.

Future work

Although they are not part of the core future framework, future-based map-reduce packages **future.apply**, **furrr**, **doFuture**, and the like, play an essential role in how developers and end-users interact with futures. A key feature of these packages is "load balancing", which helps reduce the overall overhead that comes from setting up futures and spawning them on parallel workers and collecting their results. They achieve this by partitioning the elements to iterated over into equally sized chunks, typically so that there is one chunk per worker, which in turn results in one future per

[15]The ranks are robust estimates based on the average median weekly download counts from the RStudio CRAN mirror during four weeks.

chunk and hence one future per worker. In contrast, without load balancing, each element is processed by one future resulting in more overhead, especially when there are many elements to iterate over. Each of these packages has its own implementation of load balancing, despite often using exactly the same algorithm. If there is an improvement or a bug fix to one, the maintainers of the others need to update their code too. The same is true for how they orchestrate globals and parallel RNG. To improve on this situation and to further harmonize the behavior of futures in these packages, a new helper package **future.mapreduce** that implements these common tasks will be introduced, relieving these packages from those tasks. This will also have the advantage of making it even easier to implement other types of map-reduce APIs on top of futures.

Having said this, in a longer perspective, it might be possible to remove the need for these future-based map-reduce APIs, which essentially are thin wrappers ported from their counterpart map-reduce APIs. This would require internal refactoring of the core future framework, but it can likely be done while preserving full backward compatibility with the current *Future API*. For clarification, consider the following `lapply()` construct that evaluates `slow_fcn(x)` for ten elements, each resolved via a unique *lazy* future:

```
xs <- 1:10
fs <- lapply(xs, function(x) future({
  slow_fcn(x)
}, lazy = TRUE))
```

A lazy future defers the evaluation of its expression until we use `resolved()` to query if it is resolved or until we use `value()` to collect its value[16]. Since neither has been called above, these futures are still dormant, regardless of future backend used. Next, assume that there are two parallel workers and imagine that we have a function `merge()` to merge futures. This would allow us to partition ten futures into only two futures, one per worker, and then collect their values:

```
f1 <- merge(fs[1:5])
f2 <- merge(fs[6:10])
vs <- c(value(f1), value(f2))
```

We can simplify this further by encapsulating the above in the S3 method `value()` for *list*s:

```
vs <- value(fs)
```

We can mitigate the verbosity in the setup of futures with a helper function or syntax sugar. More importantly, this would make it possible to use futures in map-reduce APIs without the need for a counterpart parallel implementation. It would also lower the threshold further for adding a thin layer of support for futures *within* existing map-reduce APIs, especially since the design of the future framework keeps the added maintenance burden to a minimum.

A frequently requested feature is to support *suspending* running futures, particularly when their runtimes are large. For example, above `future_either()` function could benefit from a `suspend()` function to terminate futures no longer needed. Since not all backends may support this, extra care needs to be taken when introducing this feature to the future framework. A related feature request is the possibility to *restart* a future that failed due to, for instance, a crashed worker or a partial power failure on a compute cluster, e.g., `restart(f)`. Combined with R's condition handling framework, higher-level APIs can then take on the role of retrying to resolve failed futures, e.g., `retry({ ... }, times = 3, on = "FutureError")`.

Implementing support for suspending and restarting futures will indirectly add support for serializing futures themselves, which is only partially supported in the current implementation. Being able to serialize futures opens up further possibilities such as saving futures to be processed at a later time, in another context, or transferring them to a job queue that, in turn, distributes them to appropriate compute resources.

The problem of not being able to export all types of objects as-is in parallel processing can be a blocker. It turns out that for a subset of these, we could use *marshaling* to encode them before serializing them such that a working clone can be reconstructed after unserializing and *unmarshaling*. As an example, a read-only file connection can be marshaled by recording its filename and file position so that the parallel worker could open its own read-only connection for the same file at the same position. Marshaling is a rarely used concept in R, possibly because there is no standard convention for package developers to rely on. Ideally, such a mechanism would allow package developers to register custom `marshal()` and `unmarshal()` methods for their data types, making them automatically applicable in parallelization without prior knowledge of what objects being transferred.

[16]Although a lazy future defers the evaluation to a later time, contrary to R's *lazy evaluation* and *promises*, a future records all dependent variables ("globals") when it is created, which means it will resolve to the same value even if those globals change after the future was created and before it was resolved. This also means that lazy and eager futures give the same value.

Other than setting the backend via `plan()`, it is not possible to direct a particular future to a specific backend type based on the needs of the future. To support this, we have to add options to declare what *resources* are needed to resolve particular future. For instance,

```
f <- future({ ... }, resources = c("r:3.2.*", "mount:/data", "!fork"))
```

could be one way to specify that this future has to be resolved with R 3.2 on a machine with a `/data` mount point and that forked parallel processing must not be used. Some resources may be implicit based on exported globals, e.g., a specific file required when exporting a file connection via marshaling.

All the above is on the roadmap for the future framework.

Summary

The **future** package is a lightweight R package that provides an alternative approach for parallel processing in R. It implements the *Future API*, which comprises three basic functions, from which richer, higher-level APIs for parallel processing can be constructed. Several of these higher-level APIs mimic counterpart map-reduce APIs closely, allowing developers to stay with their favorite coding style for their parallel needs. The future framework is designed so that the developer does not have to worry about common, critical tasks such as exporting globals to workers, using proper parallel RNG, and taking care of output, messages, warnings, and errors. This design lowers the barriers to reimplement existing algorithms and methods in parallel while avoiding increasing the maintenance burden. When using futures, the end-user controls which parallel backend is used, while the developer controls what to parallelize. This is possible because all future backends have been validated to conform to the *Future API* specifications, ensuring that futures produce the same results regardless of how and where they are processed.

Acknowledgments

I am grateful to all users and developers who have contributed to the future framework with questions and answers, feature requests, bug reports, and interesting and fruitful discussions. I am thankful to the reviewers, the editor, and everyone else who gave comments and suggestions helping to improve this article. The development of the *Future API Specifications and Conformance* test suite in the **future.tests** package was supported by the R Consortium through its Infrastructure Steering Committee (ISC) grant program.

Bibliography

H. Bengtsson. *listenv: Environments Behaving (Almost) as Lists*, 2019. URL `https://CRAN.R-project.org/package=listenv`. R package version 0.8.0. [p282]

H. Bengtsson. *globals: Identify Global Objects in R Expressions*, 2020. URL `https://CRAN.R-project.org/package=globals`. R package version 0.14.0. [p281]

H. Bengtsson. *doFuture: A Universal Foreach Parallel Adapter using the Future API of the 'future' Package*, 2021a. URL `https://CRAN.R-project.org/package=doFuture`. R package version 0.12.0. [p284]

H. Bengtsson. *future: Unified Parallel and Distributed Processing in R for Everyone*, 2021b. URL `https://CRAN.R-project.org/package=future`. R package version 1.23.0. [p273]

H. Bengtsson. *future.apply: Apply Function to Elements in Parallel using Futures*, 2021c. URL `https://CRAN.R-project.org/package=future.apply`. R package version 1.8.1. [p286]

H. Bengtsson. *future.batchtools: A Future API for Parallel and Distributed Processing using 'batchtools'*, 2021d. URL `https://CRAN.R-project.org/package=future.batchtools`. R package version 0.10.0. [p283]

H. Bengtsson. *future.callr: A Future API for Parallel Processing using 'callr'*, 2021e. URL `https://CRAN.R-project.org/package=future.callr`. R package version 0.6.1. [p283]

H. Bengtsson. *future.tests: Test Suite for Future API Backends*, 2021f. URL `https://CRAN.R-project.org/package=future.tests`. R package version 0.3.0. [p284]

H. Bengtsson. *parallelly: Enhancing the 'parallel' Package*, 2021g. URL `https://CRAN.R-project.org/package=parallelly`. R package version 1.28.1. [p284]

H. Bengtsson. *progressr: A Inclusive, Unifying API for Progress Updates*, 2021h. URL https://CRAN.R-project.org/package=progressr. R package version 0.9.0. [p280]

B. Bischl, M. Lang, O. Mersmann, J. Rahnenführer, and C. Weihs. BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments. *Journal of Statistical Software*, 64(11): 1–25, 2015. URL https://dx.doi.org/10.18637/jss.v064.i11. [p286]

W. Chang, J. Luraschi, and T. Mastny. *profvis: Interactive Visualizations for Profiling R Code*, 2020. URL https://CRAN.R-project.org/package=profvis. R package version 0.3.7. [p285]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.7.1. [p287]

J. Cheng. *promises: Abstractions for Promise-Based Asynchronous Programming*, 2021. URL https://CRAN.R-project.org/package=promises. R package version 1.2.0.1. [p281]

G. Csárdi and W. Chang. *callr: Call R from R*, 2021. URL https://CRAN.R-project.org/package=callr. R package version 3.7.0. [p283]

G. Csárdi and H. Wickham. *revdepcheck: Automated Reverse Dependency Checking*, 2021. URL https://github.com/r-lib/revdepcheck#readme. R package version 1.0.0.9001. [p284]

J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004. URL https://doi.org/10.1145/1327452.1327492. [p273]

D. Eddelbuettel. Parallel computing with R: A brief review. *WIREs Computational Statistics*, 13(2):e1515, 2021. doi: 10.1002/wics.1515. URL https://doi.org/10.1002/wics.1515. [p273]

D. Eddelbuettel, with contributions by Antoine Lucas, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, V. Simko, K. Hornik, R. Neal, K. Bell, M. de Queljoe, I. Suruceanu, and B. Denney. *digest: Create Compact Hash Digests of R Objects*, 2021. URL https://CRAN.R-project.org/package=digest. R package version 0.6.28. [p284]

M. Edmondson. *googleComputeEngineR: R Interface with Google Compute Engine*, 2019. URL https://CRAN.R-project.org/package=googleComputeEngineR. R package version 0.3.0. [p283]

D. P. Friedman and D. S. Wise. Aspects of applicative programming for parallel processing. *IEEE Transactions on Computers*, C-27(4):289–296, apr 1978. doi: 10.1109/tc.1978.1675100. URL https://doi.org/10.1109/tc.1978.1675100. [p273]

L. Henry and H. Wickham. *purrr: Functional Programming Tools*, 2020. URL https://CRAN.R-project.org/package=purrr. R package version 0.3.4. [p286]

J. Hester. *bench: High Precision Timing of R Expressions*, 2020. URL https://CRAN.R-project.org/package=bench. R package version 1.1.1. [p285]

C. Hewitt and H. G. Baker. Laws for communicating parallel processes. In *IFIP Congress*, pages 987–992, 1977. URL https://dblp.uni-trier.de/db/conf/ifip/ifip1977.html#HewittB77. [p273, 287]

P. Hibbard. Parallel processing facilities. In S. A. Schuman, editor, *New Directions in Algorithmic Languages*. IRIA, 1976. [p273]

M. Kane, J. Emerson, and S. Weston. Scalable strategies for computing with massive data. *Journal of Statistical Software*, 55(14):1–19, 2013. ISSN 1548-7660. doi: 10.18637/jss.v055.i14. URL https://doi.org/10.18637/jss.v055.i14. [p275]

G. M. Kurtzer, V. Sochat, and M. W. Bauer. Singularity: Scientific containers for mobility of compute. *PLOS One*, 12(5):e0177459, 2017. URL https://doi.org/10.1371/journal.pone.0177459. [p287]

W. M. Landau. The drake R package: A pipeline toolkit for reproducibility and high-performance computing. *Journal of Open Source Software*, 3(21), 2018. URL https://doi.org/10.21105/joss.00550. [p287]

W. M. Landau. *proffer: Profile R Code and Visualize with 'Pprof'*, 2021a. URL https://CRAN.R-project.org/package=proffer. R package version 0.1.5. [p285]

W. M. Landau. The targets R package: a dynamic make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. *Journal of Open Source Software*, 6(57):2959, 2021b. URL https://doi.org/10.21105/joss.02959. [p287]

M. Lang, B. Bischl, and D. Surmann. batchtools: Tools for R to work on batch systems. *Journal of Open Source Software*, 2(10):135, feb 2017. doi: 10.21105/joss.00135. URL https://doi.org/10.21105/joss.00135. [p283]

P. L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164, 1999. URL https://doi.org/10.1287/opre.47.1.159. [p281]

B. W. Lewis. *doRedis: 'Foreach' Parallel Adapter Using the 'Redis' Database*, 2020. URL https://CRAN.R-project.org/package=doRedis. R package version 3.0.0. [p275]

J. Luraschi, K. Kuo, K. Ushey, J. Allaire, H. Falaki, L. Wang, A. Zhang, Y. Li, and The Apache Software Foundation. *sparklyr: R Interface to Apache Spark*, 2021. URL https://CRAN.R-project.org/package=sparklyr. R package version 1.7.2. [p273]

M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, pages 3–30, 1998. URL https://doi.org/10.1145/272991.272995. [p281]

D. Merkel. Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014. [p287]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2021. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-8. [p285]

Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct*, 2020. URL https://CRAN.R-project.org/package=foreach. R package version 1.5.1. [p275]

Microsoft Corporation and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2020. URL https://CRAN.R-project.org/package=doParallel. R package version 1.0.16. [p275]

P. Miller and K. Ingersoll. *civis: R Client for the 'Civis Platform API'*, 2020. URL https://CRAN.R-project.org/package=civis. R package version 3.0.0. [p287]

M. Morgan, V. Obenchain, M. Lang, R. Thompson, and N. Turaga. *BiocParallel: Bioconductor Facilities for Parallel Evaluation*, 2021. URL https://bioconductor.org/packages/BiocParallel/. R package version 1.28.0. [p286]

Revolution Analytics and S. Weston. *doMC: Foreach Parallel Adaptor for 'parallel'*, 2020. URL https://CRAN.R-project.org/package=doMC. R package version 1.3.7. [p275]

B. Schloerke and J. Allen. *plumber: An API Generator for R*, 2021. URL https://CRAN.R-project.org/package=plumber. R package version 1.1.0. [p287]

M. Schmidberger, M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney, and U. Mansmann. State-of-the-art in parallel computing with r. *Journal of Statistical Software*, 47(1), 2009. [p273]

D. Schmidt, W.-C. Chen, M. A. Matheson, and G. Ostrouchov. Programming with BIG data in R: Scaling analytics from one to thousands of nodes. *Big Data Research*, 8:1–11, 2017. doi: https://doi.org/10.1016/j.bdr.2016.10.002. [p273]

L. Tierney. *codetools: Code Analysis Tools for R*, 2020. URL https://CRAN.R-project.org/package=codetools. R package version 0.2-18. [p281]

L. Tierney, A. J. Rossini, N. Li, and H. Sevcikova. *snow: Simple Network of Workstations*, 2021. URL https://CRAN.R-project.org/package=snow. R package version 0.4-4. [p274]

S. Urbanek. *multicore: A Stub Package to Ease Transition to 'parallel'*, 2014. URL https://CRAN.R-project.org/package=multicore. R package version 0.2. [p274]

D. Vaughan and M. Dancho. *furrr: Apply Mapping Functions in Parallel using Futures*, 2021. URL https://CRAN.R-project.org/package=furrr. R package version 0.2.3. [p286]

S. Weston. *doMPI: Foreach Parallel Adaptor for Rmpi Package*, 2017. URL https://CRAN.R-project.org/package=doMPI. R package version 0.2.2. [p275]

Henrik Bengtsson
Department of Epidemiology and Biostatistics,
University of California, San Francisco
San Francisco, CA
United States
henrik.bengtsson@gmail.com

MatchThem:: Matching and Weighting after Multiple Imputation

by Farhad Pishgar, Noah Greifer, Clémence Leyrat and Elizabeth Stuart

Abstract Balancing the distributions of the confounders across the exposure levels in an observational study through matching or weighting is an accepted method to control for confounding due to these variables when estimating the association between an exposure and outcome and reducing the degree of dependence on certain modeling assumptions. Despite the increasing popularity in practice, these procedures cannot be immediately applied to datasets with missing values. Multiple imputation of the missing data is a popular approach to account for missing values while preserving the number of units in the dataset and accounting for the uncertainty in the missing values. However, to the best of our knowledge, there is no comprehensive matching and weighting software that can be easily implemented with multiply imputed datasets. In this paper, we review this problem and suggest a framework to map out the matching and weighting of multiply imputed datasets to 5 actions as well as the best practices to assess balance in these datasets after matching and weighting. We also illustrate these approaches using a companion package for R, **MatchThem**.

1. Introduction

Researchers often seek to estimate the effect of a treatment, exposure, or policy on an outcome but may be unable to randomly assign participants into the groups to be compared. The inability to randomize can lead to differences between the distributions of participant characteristics between exposure groups (known as *covariate imbalance*), which is the source of confounding bias in a naïve estimate of the exposure effect. When enough confounders—causes of exposure status and the outcome of interest—have been observed, one strategy for reducing this bias is to equate the confounder distributions across the exposure groups by matching or weighting units prior to estimating the exposure effect. Ideally, after matching or weighting, the exposure groups will be *balanced*, and a simple or covariate-adjusted estimate of the difference in average outcomes between the exposure groups will be unbiased for the true exposure effect (Stuart, 2010). Matching and weighting can also enhance the robustness to misspecification of any outcome models used to estimate the exposure effect (Ho et al., 2007).

Despite increasing popularity in practice, matching and weighting methods cannot be immediately applied to datasets with missing values. There are several solutions to address the problem of missing data in causal effect estimation, but a standard and relatively easy-to-use strategy is multiple imputation of the missing data, which preserves the number of units in the dataset while accounting for some of the uncertainty in the missing values (Cham and West, 2016). Multiple imputation involves filling in the missing data points using estimates of their values, repeating the process multiple times with randomness incorporated into each prediction to arrive at a set of multiple complete datasets containing the imputed values. The difficulty of analyzing multiply imputed data is that any analysis must be carried out within each imputed dataset, and the results pooled together using specific combining rules to arrive at a single set of estimates. Because matching and weighting are iterative, multi-step procedures, it is not straightforward to implement an analysis using these methods without extensive programming.

The **MatchThem** R package, which we introduce here, offers an analysis pipeline for estimating exposure effects using matching and weighting with multiply imputed data. The functions **MatchThem** offers blend seamlessly with functions used in other R packages for matching, weighting, and the generation and analysis of multiply imputed data. The aims of the present paper are to briefly review the issues around matching and weighting with multiply imputed data (section 2), to describe the structure and functionality of **MatchThem** (section 3), to describe the steps involved in implementing the best practices for these procedures (section 4), and to demonstrate the typical use of the **MatchThem** R package (section 5).

1.1. Notation

Let $i = 1, 2, 3, ..., n$ index the n units in a dataset, in which the causal effects of a binary exposure indicator (z) on an outcome (y) in the presence of a set of potential confounders ($X = \{x_1, x_2, x_3, ...\}$) are to be estimated (such that $z_i = 0$ indicates that unit i is assigned to the control group and $z_i = 1$ indicates that the unit i is assigned to the treated group) (Figure 1).

The typical context in which matching and weighting are used is one where data have been collected from an observational study in which the exposure is not randomized, yielding systematic differences between exposed and unexposed units on a set of measured potential confounders (often referred to as *covariates*). The situation we consider here is one in which the values of some of the covariates or the outcome are missing for some units in the observed dataset (we do not consider situations in which the exposure status is missing as the methods described herein may not apply to such scenarios). In order to account for the missingness in the covariates and outcome, the missing values are multiply imputed, creating m complete datasets. Though we briefly explain the procedure of multiple imputation in section 2, here, we focus on the procedures following imputation; see (White et al., 2011) and (Azur et al., 2011) for accessible introductions to multiple imputation for medical researchers.

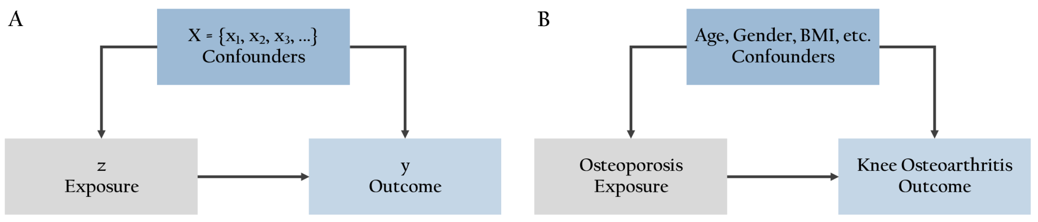


Figure 1: The Research Question. A. The notations used in this paper. B. The research question used as an example in this paper.

1.2. Software requirements

The **MatchThem** package works with the R statistical software and programming language and can be installed in R (with version $\geqslant$ 3.6.0) running on different platforms. **MatchThem** can be installed from the Comprehensive R Archive Network by executing the following commands in the R software console (the **MatchThem** package depends on the **MatchIt** (Ho et al., 2011) and **WeightIt** (Greifer, 2020a) packages; these lines will install those packages too):

```
install.packages("MatchThem")
library(MatchThem)
```

2. Matching, weighting, and missing data

2.1. Matching

Matching and weighting are methods to equate the distributions of the covariates between exposure groups (Stuart, 2010). Matching does so by duplicating, selecting, or dropping units from the dataset in such a way that the resulting exposure groups have similar covariate distributions. Typically, matching relies on a distance measure constructed from the covariates to pair similar units between exposure groups, which then form the resulting matched sample; a popular distance measure is the propensity score difference, the propensity score being the predicted probability of being in the exposed group given the covariates. Propensity scores can be used in a variety of matching algorithms (Ho et al., 2011; Williamson et al., 2012), though other distance measures can be used as well (as there have been some recent concerns about the use of propensity scores for matching (King and Nielsen, 2019)). Matching produces a set of matching weights (often 1 for those retained and 0 for those dropped) and matched pair membership, which can be incorporated into a regression of the outcome on the exposure to estimate the exposure effect in the matched sample. If the balance is achieved across the exposure groups in the matched sample, then bias in the exposure effect estimate will be reduced.

Matching is implemented in a number of R packages, but the **MatchIt** package provides access to a variety of matching methods for complete (i.e., non-missing) data. The **MatchIt** function `matchit()` performs the requested form of matching on the supplied dataset, returning an object from which matching weights and matched pair membership can be extracted for use in effect estimation. **MatchThem** interfaces with **MatchIt** to extend `matchit()` for use with multiply imputed data.

2.2. Weighting

Weighting is another way to achieve balance and reduce bias in the estimate of an exposure effect. Weights for each unit can be estimated so that the distributions of the covariates are the same across the exposure groups in the weighted samples. The weights then function like survey weights and can be used in a weighted regression of the outcome on the exposure to estimate the exposure effect. A common way of estimating weights is to use a function of the estimated propensity score, a procedure known as inverse probability weighting (IPW), though there have been some developments that bypass estimating the propensity score to estimate the weights directly (Hainmueller, 2012; Zubizarreta, 2015).

The R package **WeightIt** implements a variety of weighting methods and functions similarly to **MatchIt**. The **WeightIt**'s function, `weightit()`, performs the requested form of weighting and returns an object containing the estimated weights. **MatchThem** interfaces with **WeightIt** to extend `weightit()` for use with multiply imputed data.

2.3. Assessing covariate balance

After matching or weighting, one must assess the degree to which the balancing method was successful at achieving covariate balance in the exposure groups. This involves using numerical and graphical criteria to compare the distributions of covariates across the groups (Austin, 2009). If the balance is not achieved, the matching or weighting specification should be changed, and the procedure performed again until a satisfactory balance is found. The **cobalt** package provides tools for assessing balance after matching and weighting and has tools for summarizing balance in multiply imputed data. **MatchThem** interfaces with **cobalt** to facilitate balance checking as part of a complete analysis pipeline. We refer readers to the **cobalt** documentation for further explanation of **cobalt**'s capabilities in order to maintain focus on the structure and functionality of **MatchThem**, but we will include the use of **cobalt** in the demonstration of the analysis pipeline in section 5.

2.4. Missing data

A major obstacle for most matching and weighting procedures is that they cannot be performed in a straightforward way for units with missing values in the covariates because these procedures either search control and treat groups for units with similar covariate values or rely on the predictions from a model with the covariates as the predictors (i.e., the propensity scores), which cannot be computed in the presence of missing data.

Complete-case analysis, i.e., excluding units with missing values in the potential confounders or outcome, is a simple and naïve approach for handling missing data. However, the complete-case analysis may not be a valid option in all instances; the assumption of missingness completely-at-random, described below, is required to justify complete-case analysis and is often violated in observational data. In addition, it is possible that dropping units with any missing values may yield a dataset with few remaining units (Pigott, 2001). Another approach is to replace the missing values with an arbitrary constant and include indicators for missingness as additional covariates in X, though this can also allow bias to remain (Knol et al., 2010). The preferred method to address the problem of missing data that preserves the number of units in the dataset and often yields unbiased effect estimates is to impute the missing values using multiple imputation (Leyrat et al., 2019).

Multiple imputation refers to the procedure of substituting the missing values with a set of plausible values that reflect the uncertainty in predicting the true unobserved values, which results in m imputed (filled-in) datasets (Sterne et al., 2009). Multiple imputation is justified when the mechanism behind the missingness is ignorable, i.e., given the observed data, units with missing data represent a random subset of the dataset ('missing-completely-at-random' in Rubin's language (Rubin, 1987)) or when the probability that a value is missing relies on values of other observed variables, but not on the missing value itself or unobserved factors ('missing-at-random' in Rubin's language (Rubin, 1987)).

Several multiple imputation methods are described in the literature, and multiple statistical packages can be used to generate multiply imputed data. The general framework of these methods is the same: impute the missing values to produce m datasets, analyze the imputed datasets separately, and pool the results obtained in each imputed dataset using standard combining rules to arrive at a single estimate for the sample (Sterne et al., 2009; Rubin, 1987). A popular method of multiple imputation is multiple imputation with chained equations (MICE), which involves iteratively fitting models to predict the missing values and is implemented in the **mice** R package. The **mice** package contains the functions `mice()` to impute the missing values, `with()` to run a supplied analysis model on each imputed dataset, and `pool()` to pool the results of the models to arrive at a single set of coefficient estimates and standard errors, facilitating the creation and analysis of multiply imputed data in a single analysis pipeline requiring minimal programming. We refer the reader to the **mice** documentation for further details (van Buuren and Groothuis-Oudshoorn, 2011).

2.5. Matching and weighting multiply imputed datasets

Given the limitations of conducting a complete-case analysis, multiply imputing missing data before matching or weighting has become a popular alternative for use with missing data. There has been some research examining the performance and optimal use of matching and weighting with multiply imputed data, with a focus on the correct sequence of actions involved. There are two main approaches that have been identified:

1. The *within* approach: In this approach, matching or weighting is performed within each imputed dataset, using the observed and imputed covariate values, and the exposure effects estimated in each of the m matched or weighted datasets are pooled together (Leyrat et al., 2019).
2. The *across* approach: In this approach, propensity scores are averaged across the imputed datasets, and, using this averaged measure, matching or weighting is performed in the imputed datasets. Finally, the estimated exposure effects obtained from analyzing the matched or weighted datasets are pooled together (Mitra and Reiter, 2016).

The across approach has been demonstrated to have inferior statistical performance as compared to the within approach in many common scenarios (Leyrat et al., 2019; de Vries and Groenwold, 2016), though early research favored its use (Mitra and Reiter, 2016). In particular, the across approach seems most effective when the outcomes are not used to impute the missing covariate values (de Vries and Groenwold, 2016). Although some recommend avoiding the inclusion of the outcome variable during or prior to matching and weighting with propensity scores (Rubin, 2001), statistical evidence favors the use of the outcome variable in multiple imputation of covariates (Leyrat et al., 2019). In addition, the across approach is not compatible with matching and weighting methods that do not involve propensity scores, such as coarsened exact matching (de Vries and Groenwold, 2016), Mahalanobis distance matching, and entropy balancing (Hainmueller, 2012). Both approaches are implemented in **MatchThem** to facilitate comparison between them, though the within approach is the default in **MatchThem** functions and is the approach we recommend.

It should be noted that the across approach described by Mitra and Reiter (2016) differs slightly from that described here; in their procedure, the averaged propensity scores are used to estimate the causal effect in a single dataset consisting of just the observed exposure and outcome values, which are assumed to be non-missing. The procedure described here is in the spirit of the original method but allows for the presence of imputed outcomes and the use of imputed covariates in the effect estimation. When there is no missingness in the outcome and covariates are not used in the effect estimation, the two versions of this approach coincide.

3. Package contents and structure

MatchThem provides functions and `S3` classes to facilitate the use of matching and weighting with multiply imputed data and the estimation of exposure effects and their uncertainty (i.e., standard errors), which requires special care when done with matched or weighted multiply imputed data. In particular, **MatchThem** extends the functionality of **MatchIt** and **WeightIt** for matching and weighting to multiply imputed data and the functionality of **mice** for the analysis of multiply imputed data to matched and weighted data. **MatchThem** provides wrappers for functions in these packages to create a smooth workflow requiring minimal programming. Table 1 contains a summary of the functions and classes contained in **MatchThem**.

The **MatchThem** functions `matchthem()` and `weightthem()` are wrappers for `MatchIt::matchit()` and `WeightIt::weightit()` that apply them to each imputed dataset, supplied in the form of a `"mids"` object, the output of a call to `mice::mice()`, which contains the multiply imputed datasets (`matchthem()` and `weightthem()` are also compatible with `"amelia"` objects from the **Amelia** package, but they are first transformed into `"mids"` objects before matching or weighting is performed on them). `matchthem()` and `weightthem()` apply the corresponding functions to the imputed datasets using the requested approach, storing the outputs along with the original imputed data in a `"mimids"` or `"wimids"` object, which extend `mice`'s `"mids"` class to additionally contain the matching and weighting output. The `"mimids"` and `"wimids"` classes have a number of methods that extend `mice`'s functions for analyzing `"mids"` objects; in particular, **MatchThem** offers `complete()`, `with()`, and `pool()`, which function similarly to their equivalents in **mice**. **MatchThem** also contains methods for `cbind()`, `print()`, `summary()`, and `plot()` with `"mimids"` and `"wimids"` objects. We describe the syntax of these functions below.

Function	Input	Output	Extends	Description
`matchthem()`	`mids` object	`mimids` object	`MatchIt::matchit()`	Performs the requested form of matching on the imputed datasets.
`weightthem()`	`mids` object	`wimids` object	`WeightIt::weightit()`	Performs the requested form of weighting on the imputed datasets.
`complete()`	`mimids` or `wimids` object	`data.frame`[1]	`mice::complete()`	Extracts one or more imputed datasets from the supplied input along with the outputs of the matching or weighting procedure.
`with()`	`mimids` or `wimids` object	`mimira` object	`mice::with()`	Runs the supplied analysis model on each imputed dataset, incorporating the outputs of the matching or weighting procedure.
`pool()`	`mimira` object	`mimipo` object[2]	`mice::pool()`	Pools the coefficients and standard errors estimated across the imputed datasets to a single set of coefficient and standard error estimates.

Table 1: Primary Functions in **MatchThem**. 1. `complete()` can also produce outputs in other forms. 2. `mimipo` objects can be further analyzed by functions in **mice** as if they had come from `mice::pool()`.

3.1. matchthem()

The syntax for matchthem() is as follows:

```
matchthem(formula, datasets,
          approach = "within",
          method = "nearest",
          distance = "logit",
          distance.options = list(),
          discard = "none",
          reestimate = FALSE, ...)
```

The formula argument corresponds to the model formula, which relates the exposure (on the left-hand side) to the covariates. The datasets argument corresponds to the "mids" or "amelia" object containing the multiply imputed datasets. The approach argument, with options "within" and "across", corresponds to the approach used. The method argument corresponds to the method of matching used, which, as of now, can be one of the nearest neighbor matching ("nearest"), optimal full matching ("full"), propensity score subclassification ("subclass"), optimal pair matching ("optimal"), exact matching ("exact"), coarsened exact matching ("cem"), and genetic matching ("genetic"). The distance argument corresponds to the method used to define the distances between units; it can be "mahalanobis" for Mahalanobis distance matching or a method of estimating propensity scores (the default, "glm", estimates propensity scores using logistic regression). Note that only the methods that involve propensity scores are allowed with the across approach; as of now, these include "nearest", "full", "subclass", "optimal", and "genetic", and only when propensity scores are requested. The other arguments, including those supplied to ..., control other aspects of the matching procedure and are, along with formula, method, and distance, passed directly to matchit().

3.2. weightthem()

The syntax for weightthem() is as follows:

```
weightthem(formula, datasets,
           approach = "within",
           method = "ps", ...)
```

The formula, datasets, and approach arguments have the same meaning as those for matchthem(). The method argument controls the weighting method used, which, as of now, can be one of logistic regression propensity score weighting ("ps"), covariate balancing propensity score weighting ("cbps") and its nonparametric variety ("npcbps"), generalized boosted modeling propensity score weighting ("gbm"), entropy balancing ("ebal"), SuperLearner propensity score weighting ("super"), optimization-based weighting ("optweight"), energy balancing ("energy"), and Bayesian additive regression tree propensity score weighting ("bart"). Note that only methods that involve propensity scores can be used with the across approach; as of now, these include "ps", "cbps", "gbm", "super", and "bart". Arguments supplied to ... are passed to weightit() to control details of the weight estimation.

3.3. complete()

The syntax for complete.mimids() is as follows (the syntax for complete.wimids() is identical):

```
complete.mimids(data, action = 1,
                include = FALSE, mild = FALSE, all = TRUE, ...)

complete.wimids(data, action = 1,
                include = FALSE, mild = FALSE, all = TRUE, ...)
```

complete() extracts the multiply imputed and matched or weighted datasets from a "mimids" or "wimids" object, yielding a "data.frame" containing the imputed data and the outputs of the matching or weighting function. These additional outputs include the estimated weights (which are produced with both matching and weighting), matched pair membership, and estimated propensity scores (if used). This function extends complete.mids() from the **mice** package. The data argument corresponds to a "mimids" or "wimids" object, the output of a call to matchthem() or weightthem(). The action argument controls the format of the output; it can be supplied as a number to extract a single dataset corresponding to that imputation number or a string, such as "long", to produce an object (of one of several types) containing all of the imputed datasets; the mild argument provides another way to control output. The `all` argument controls whether all units should be included in the output or just units with a weight greater than zero (i.e., units that have not been discarded or that were left unmatched). Though the datasets produced by `complete()` can be analyzed separately and the estimates manually combined using the appropriate combining rules, the `with()` and `pool()` methods facilitate proper analysis of the matched or weighted imputed data.

3.4. `with()`

The syntax for `with.mimids()` and `with.wimids()` are as follows:

```
with.mimids(data, expr, cluster, ...)

with.wimids(data, expr, ...)
```

The `with()` method for `"mimids"` and `"wimids"` objects extends the `with()` method for `"mids"` objects provided in **mice**. It works by extracting the imputed datasets one-by-one along with the matching or weighting outputs and applies the modeling function supplied to `expr` (e.g., a call to `glm()`, `survey::svyglm()`, or `survival::coxph()` with the outcome model formula included) to each of the datasets. No data argument needs to be supplied to the modeling function because the imputed datasets are automatically supplied by `with()`. `with()` automatically incorporates the estimated weights in the estimation function when possible. The output of a call to with is a `"mimira"` object, which contains the outputs of the models run on each imputed dataset and extends the corresponding `"mira"` class from **mice**.

In general, for generalized linear outcome models, the `svyglm()` function in the **survey** package should be used as it correctly accounts for the weights and produces approximately correct standard errors, whereas the standard errors resulting from a normal call to `glm()` will be inaccurate. `with()` automatically constructs and supplies the `svydesign` object containing the data and weights, so neither need to be supplied. When matching is used, and a modeling function from the **survey** package is supplied, information about pair membership is also supplied to appropriately account for the clustering in the standard error estimation as recommended by (Austin, 2011) (this functionality can be controlled using the cluster argument).

3.5. `pool()`

The syntax for `pool()` is as follows:

```
pool(object, dfcom = NULL)
```

`pool()` takes in a `"mimira"` object (supplied to the `object` argument) to pool the models and provide a single set of coefficient estimates and information required to compute their standard errors. The `dfcom` argument controls the degrees of freedom used for the tests of the coefficients and confidence intervals, which typically is close to the number of units in the original dataset. Because matching and weighting can yield datasets with different numbers of units remaining, the default is to use the smallest degrees of freedom from the supplied models if possible; otherwise, a large value is used to approximate a z-test. **MatchThem** re-exports `pool()` as a generic with methods for `"mira"` objects (the output of `mice::with.mids()`) and `"mimira"` objects as `mice::pool()` is not a generic. The output of `MatchThem::pool.mimira()` is a `"mimipo"` object, which can be used with the methods available in **mice** for `"mipo"` objects (e.g., `summary()` and `print()`).

3.6. Methods for `"mimids"` and `"wimids"` objects

MatchThem also contains methods for `cbind()`, `print()`, `summary()`, and `plot()` with `"mimids"` and `"wimids"` objects. `cbind()` allows one to add variables from an external dataset, not included in the original `"mids"` object, that one might wish to be involved in the effect estimation model, such as an outcome not involved in the imputation or a variable collected after the multiple imputation occurred. The `print()`, `summary()`, and `plot()` methods simply apply the corresponding function to the `matchit` or `weightit` objects contained within the `"mimids"` or `"wimids"` object. The **MatchIt** and **WeightIt** documentation details their functionality. An additional argument, `n`, determines to which imputed dataset the function should be applied (e.g., `print.mimids(x,n = 1)` prints the output of the call to `matchit()` used on the first imputed dataset).

4. Suggested workflow

The suggested workflow for pre-processing imputed datasets with matching or weighting and then analyzing them to estimate exposure effects using **MatchThem** is as follows (Figure 2):

1. **Imputing the Missing Data in the Dataset**: Data are imputed using functions in **mice** or **Amelia**. Data imputed using another package can be coerced to a `"mids"` object by the **mice** function `as.mids()` for use with **MatchThem** functions (van Buuren and Groothuis-Oudshoorn, 2011; Honaker et al., 2011).
2. **Matching or Weighting the Imputed Datasets**: Matching or weighting are performed using `matchthem()` or `weightthem()`, respectively, on the imputed datasets. The functions perform the matching or weighting within each imputed dataset using the specified approach.
3. **Assessing Balance on the Matched or Weighted Datasets**: Functions in the **cobalt** R package can be used to assess the balance to ensure that the resulting bias is small across imputed datasets. The `bal.tab()` and `love.plot()` functions in the **cobalt** package can be used directly on the output of `matchthem()` and `weightthem()` (Greifer, 2020b). If the balance is not achieved, step 2 should be repeated with different approaches or methods until it is.
4. **Analyzing the Matched or Weighted Datasets**: Using `with()` function from **MatchThem** package, causal effects and their standard errors are estimated in each of the matched or weighted imputed datasets. Robust standard errors should be used with weighting and most matching methods and are available through integration with the **survey** package (Lumley, 2004).
5. **Pooling the Causal Effect Estimates**: The `pool()` function from the package is used to pool the obtained causal effect estimates and standard errors from each dataset using Rubin's rules.

5. Example

In this section, we review the suggested workflow for matching and weighting multiply imputed datasets, using an example. The research question in this context is whether osteoporosis at baseline is associated with increased odds of developing knee osteoarthritis in the follow-up or not (Figure 1). We will use the `osteoarthritis` dataset (included in the **MatchThem** package):

```
library(MatchThem)
data('osteoarthritis')
```

The `osteoarthritis` dataset contains data on 7 characteristics (age: `AGE`, gender: `SEX`, body mass index: `BMI`, racial background: `RAC`, smoking status: `SMK`, osteoporosis at baseline: `OSP`, and knee osteoarthritis in the follow-up: `KOA`) of 2,585 individuals. The dataset contains missing data in `BMI`, `RAC`, `SMK`, and `KOA` variables. We assume the missing values are missing at random, justifying the use of multiple imputation.

```
summary(osteoarthritis)
```

5.1. Imputing the missing data in the dataset

We use the **mice** package to impute the missing data in the `osteoarthritis` dataset. See the **mice** package reference manual for more details about this step (van Buuren and Groothuis-Oudshoorn, 2011). We use 5 imputations for illustration, though more imputations are always better.

```
library(mice)
imputed.datasets <- mice(osteoarthritis, m = 5)
```

The code above produces the 5 imputed datasets and saves them in the `imputed.datasets` object (of class `"mids"`). This object will be supplied to **MatchThem** functions to perform matching and weighting in the imputed datasets.

5.2. Matching and weighting the imputed datasets

5.2.1. Matching the imputed datasets

In this example, we use `matchthem()` to match the multiply imputed datasets, `imputed.datasets`, using the `"within"` matching approach with nearest neighbor matching on the propensity score,

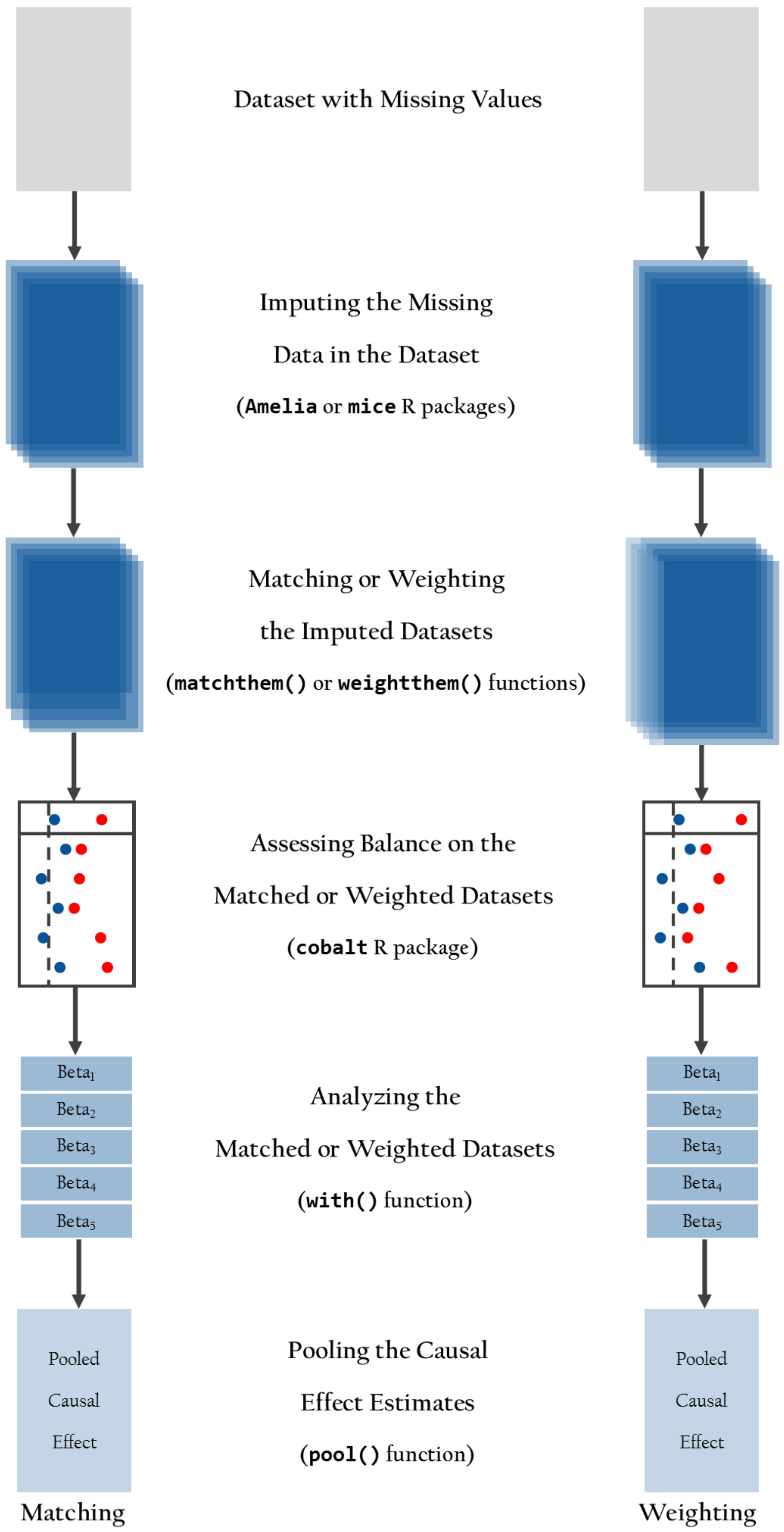

Dataset with Missing Values
Imputing the Missing Data in the Dataset (Amelia or mice R packages)
Matching or Weighting the Imputed Datasets (matchthem() or weightthem() functions)
Assessing Balance on the Matched or Weighted Datasets (cobalt R package)
Analyzing the Matched or Weighted Datasets (with() function)
Pooling the Causal Effect Estimates (pool() function)
Beta1
Beta2
Beta3
Beta4
Beta5
Pooled Causal Effect
Matching
Weighting

Figure 2: Suggested Workflow for Matching and Weighting Multiply Imputed Datasets

a caliper of 5% of the standard deviation of the propensity score, and 2:1 unexposed-to-exposed matching ratio for matching. The syntax is the same as it is for `MatchIt::matchit()`, except that the `imputed.datasets` is supplied to the `datasets` argument (whereas `matchit()` takes a "`data.frame`" for its `data` argument) and an argument to `approach` is supplied to select the approach to be used.

```
matched.datasets <- matchthem(OSP ~ AGE + SEX + BMI + RAC + SMK,
                              datasets = imputed.datasets,
                              approach = 'within',
                              method = 'nearest',
                              caliper = 0.05,
                              ratio = 2)

# Matching Observations  | dataset: #1 #2 #3 #4 #5
```

After the matching is performed in the 5 imputed datasets, the output will be saved in the `matched.datasets` object (of "`mimids`" class). The "`mimids`" object contains the original imputed data and the output of the calls to `matchit()` applied to each imputed dataset.

5.2.2. Weighting the imputed datasets

Here, we use `weightthem()` to weight the imputed datasets, `imputed.datasets`, using the "`across`" weighting approach with logistic regression propensity score weighting targeting the average treatment effect in the matched sample (ATM) estimand (which mimics the target population resulting from matching with a caliper (Li and Greene, 2013)). The syntax is the same as it is for `WeightIt::weightit()` except that the `imputed.datasets` is supplied to the `datasets` argument (whereas `weightit()` takes a "`data.frame`" for its `data` argument) and an argument to `approach` is supplied to select the approach to be used.

```
weighted.datasets <- weightthem(OSP ~ AGE + SEX + BMI + RAC + SMK,
                                datasets = imputed.datasets,
                                approach = 'across',
                                method = 'ps',
                                estimand = 'ATM')

# Estimating distances   | dataset: #1 #2 #3 #4 #5
# Estimating weights     | dataset: #1 #2 #3 #4 #5
```

The `weighted.datasets` object (of "`wimids`" class) contains the original imputed data and the output of the calls to `weightit()` applied to each imputed dataset.

5.3. Assessing balance on the matched and weighted datasets

Functions in the **cobalt** package are compatible with "`mimids`" and "`wimids`" objects, and the degree to which balance was achieved in the matched and weighted datasets of these objects can be assessed using the **cobalt** functions `bal.tab()`, `bal.plot()`, and `love.plot()`. Here, we illustrate the use of `bal.tab()` to compute absolute standardized mean differences (ASMDs) and Kolmogorov-Smirnov (KS) statistics for each covariate. The code below produces the largest ASMD and KS statistic after matching for each covariate across all the imputed datasets, indicating the worst balance across the datasets (Greifer, 2020b).

```
library(cobalt)
bal.tab(matched.datasets, stats = c('m', 'ks'),
        imp.fun = 'max')

# Balance summary across all imputations
#              Type Max.Diff.Adj Max.KS.Adj
# distance Distance       0.0107     0.0300
# AGE       Contin.       0.0296     0.0450
# SEX_2      Binary       0.0021     0.0021
# BMI       Contin.       0.0333     0.0557
# RAC_0      Binary       0.0021     0.0021
# RAC_1      Binary       0.0118     0.0118
# RAC_2      Binary       0.0139     0.0139
# RAC_3      Binary       0.0021     0.0021
# SMK        Binary       0.0128     0.0128

# Average sample sizes across imputations
#                      Control Treated
# All                  2106.     479.
# Matched (ESS)         738.17   467.2
# Matched (Unweighted)  810.2    467.2
# Unmatched            1295.8     11.8
```

This information shows that the covariates are well-balanced in the osteoporosis negative and positive groups after matching as the largest ASMD and KS statistics for all covariates across the imputed datasets are close to zero. The sample size information below indicates that some units were left unmatched and dropped from the sample. The displayed values are averaged across the imputed datasets.

We can produce the same balance table for the weighted datasets:

```
bal.tab(weighted.datasets, stats = c('m', 'ks'),
        imp.fun = 'max')

# Balance summary across all imputations
#                Type Max.Diff.Adj Max.KS.Adj
# prop.score Distance       0.0040     0.0433
# AGE         Contin.       0.0209     0.0376
# SEX_2        Binary       0.0002     0.0002
# BMI         Contin.       0.0070     0.0466
# RAC_0        Binary       0.0002     0.0002
# RAC_1        Binary       0.0019     0.0019
# RAC_2        Binary       0.0027     0.0027
# RAC_3        Binary       0.0006     0.0006
# SMK          Binary       0.0138     0.0138

# Average effective sample sizes across imputations
#            Control Treated
# Unadjusted 2106.     479.
# Adjusted    954.44   478.3
```

As with the matched datasets, the covariates are well balanced in the weighted datasets as demonstrated by the low values of the largest ASMD and KS statistics across the datasets. For more information on the available options for assessing balance with multiply imputed data, we refer the reader to the **cobalt** documentation (Greifer, 2020b).

5.4. Analyzing the matched and weighted datasets

The exposure effect within each imputed dataset can be estimated using `with()`, which applies the supplied outcome model to each of the matched or weighted datasets and stores the output in a `"mimira"` object. We illustrate the use of `with()` below to estimate the difference in the log odds of knee osteoarthritis between osteoporosis groups in the matched imputed datasets:

```
library(survey)
matched.models <- with(matched.datasets,
                       svyglm(KOA ~ OSP, family = quasibinomial()),
                       cluster = TRUE)
```

The models fit in each matched dataset are saved in the `matched.models` object (of `"mimira"` class). We can run the same code with the weighted imputed datasets:

```
weighted.models <- with(weighted.datasets,
                        svyglm(KOA ~ OSP, family = quasibinomial()))
```

Results are saved in the `weighted.models` object (of `"mimira"` class, note that in the calls to `svyglm()`, no `svydesign()` or `weights` arguments need to be specified as these are automatically supplied by `with()`)

5.5. Pooling the causal effect estimates

In order to arrive at a single set of coefficient and standard error estimates from the imputed datasets, we must pool the estimated models using `pool()`. We demonstrate this below on the `matched.models` object containing the models we fit above.

```
matched.results <- pool(matched.models)
```

The output of the `pool()` is saved in the `matched.results` object, which is of `"mimipo"` class. We can run `summary()` to arrive at a final set of estimates for the matched data:

```
summary(matched.results, conf.int = TRUE)

#          term   estimate  std.error  statistic       df  p.value    2.5 %    97.5 %
# 1 (Intercept) -0.2045680 0.08781317 -2.3295816 47.88960 0.024092 -0.38114 -0.027997
# 2        OSP1 -0.1255041 0.14138543 -0.8876733 53.09776 0.378720 -0.40908  0.158067
```

The displayed results show that there is not sufficient evidence of an association between osteoporosis and knee osteoarthritis development in the follow-up in this sample (beta = -0.13 [-0.41 – 0.16], odds ratio = 0.88 [0.66 – 1.17]).

We can run `pool()` and then `summary()` on the model fits in the weighted datasets to arrive at a similar table of results:

```
weighted.results <- pool(weighted.models)
summary(weighted.results, conf.int = TRUE)

#          term   estimate  std.error statistic      df    p.value    2.5 %    97.5 %
# 1 (Intercept) -0.1602278 0.06932782 -2.311162 225.930 0.02172525 -0.29684 -0.023616
# 2        OSP1 -0.1817342 0.13059573 -1.391579  63.768 0.16888557 -0.44265  0.079179
```

Again, there is no evidence for an association between osteoporosis and knee osteoarthritis development in this sample.

6. Summary

Matching and weighting are popular methods used to balance the distributions of potential confounders across the exposure levels in an observational study. However, these procedures cannot be immediately applied to datasets with missing values. Multiple imputation of the missing data can be an effective approach to account for missing values while preserving the number of units in the dataset and accounting for the uncertainty in the imputation of the missing values. In this paper, we described the functionality of **MatchThem**, which interfaces with **MatchIt**, **WeightIt**, and **mice** to provide a smooth, straightforward workflow for estimating exposure effects in multiply imputed data using matching or weighting. **MatchThem** contains functions and classes that encourage the use of best practices in analyzing matched or weighted multiply imputed data without requiring extensive programming by the user. Given the ubiquity of missing data in observational studies, we hope **MatchThem** will facilitate smart choices and reliable analyses by researchers attempting to estimate causal effects from observational data.

Bibliography

P. C. Austin. Some methods of propensity-score matching had superior performance to others: Results of an empirical investigation and monte carlo simulations. *Statistics in Medicine*, 51(1):171–184, 2009. URL `https://doi.org/10.1002/sim.4200`. [p294]

P. C. Austin. Comparing paired vs non-paired statistical methods of analyses when making inferences about absolute risk reductions in propensity-score matched samples. *Statistics in Medicine*, 30(11): 1292–1301, 2011. URL `https://doi.org/10.1002/sim.4200`. [p298]

M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf. Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49, 2011. URL `https://doi.org/10.1002/mpr.329`. [p293]

H. Cham and S. G. West. Propensity score analysis with missing data. *Psychological Methods*, 21(3): 427–445, 2016. URL `https://doi.org/10.1037/met0000076`. [p292]

B. P. de Vries and R. Groenwold. Comments on propensity score matching following multiple imputation. 25(6):3066—-3068, 2016. URL https://doi.org/10.1177/0962280216674296. [p295]

N. Greifer. *WeightIt: Weighting for Covariate Balance in Observational Studies*, 2020a. URL https://CRAN.R-project.org/package=WeightIt. R package version 0.9.0. [p293]

N. Greifer. *cobalt: Covariate Balance Tables and Plots*, 2020b. URL https://CRAN.R-project.org/package=cobalt. R package version 4.2.2. [p299, 301, 302]

J. Hainmueller. Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies. *Political Analysis*, 20(1):25–46, 2012. URL https://doi.org/10.1093/pan/mpr025. [p294, 295]

D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political Analysis*, 15(3):199–236, 2007. URL https://doi.org/10.1093/pan/mpl013. [p292]

D. E. Ho, K. Imai, G. King, and E. A. Stuart. MatchIt: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, 42(8):1–28, 2011. URL https://doi.org/10.18637/jss.v042.i08. [p293]

J. Honaker, G. King, and M. Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011. URL https://doi.org/10.18637/jss.v045.i07. [p299]

G. King and R. Nielsen. Why propensity scores should not be used for matching. *Political Analysis*, 27 (4):435–454, 2019. URL https://doi.org/10.1017/pan.2019.11. [p293]

M. J. Knol, K. J. M. Janssen, A. R. T. Donders, A. C. G. Egberts, E. R. Heerdink, D. E. Grobbee, K. G. M. Moons, and M. I. Geerlings. Unpredictable bias when using the missing indicator method or complete case analysis for missing confounder values: An empirical example. *Journal of Clinical Epidemiology*, 63(7):728–736, 2010. URL https://doi.org/10.1016/j.jclinepi.2009.08.028. [p294]

C. Leyrat, S. R. Seaman, I. R. White, I. Douglas, L. Smeeth, J. Kim, M. Resche-Rigon, J. R. Carpenter, and E. J. Williamson. Propensity score analysis with partially observed covariates: How should multiple imputation be used? *Statistical Methods in Medical Research*, 28(1):3–19, 2019. URL https://doi.org/10.1177/0962280217713032. [p294, 295]

L. Li and T. Greene. A weighting analogue to pair matching in propensity score analysis. *The International Journal of Biostatistics*, 9(2):215–234, 2013. URL https://doi.org/10.1515/ijb-2012-0030. [p301]

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software, Articles*, 9(8):1–19, 2004. URL https://doi.org/10.18637/jss.v009.i08. [p299]

R. Mitra and J. P. Reiter. A comparison of two methods of estimating propensity scores after multiple imputation. *Statistical Methods in Medical Research*, 25(1):188–204, 2016. URL https://doi.org/10.1177/0962280212445945. [p295]

T. D. Pigott. A review of methods for missing data. *Educational Research and Evaluation*, 7(4):353–383, 2001. URL https://doi.org/10.1076/edre.7.4.353.8937. [p294]

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*, volume 81. John Wiley & Sons, 1987. [p294]

D. B. Rubin. Using propensity scores to help design observational studies: Application to the tobacco litigation. *Health Services and Outcomes Research Methodology*, 2(3-4):169–188, 2001. URL https://doi.org/10.1023/A:1020363010465. [p295]

J. A. Sterne, I. R. White, J. B. Carlin, M. Spratt, P. Royston, M. G. Kenward, A. M. Wood, and J. R. Carpenter. Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls. *BMJ*, 338, 2009. URL https://doi.org/10.1136/bmj.b2393. [p294]

E. A. Stuart. Matching methods for causal inference: A review and a look forward. *Statistical Science*, 25(1):1–21, 2010. URL https://doi.org/10.1214/09-STS313. [p292, 293]

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. URL https://doi.org/10.18637/jss.v045.i03. [p294, 299]

I. R. White, P. Royston, and A. M. Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Statistics in Medicine*, 30(4):377–399, 2011. URL https://doi.org/10.1002/sim.4067. [p293]

E. Williamson, R. Morley, A. Lucas, and J. Carpenter. Propensity scores: From naive enthusiasm to intuitive understanding. *Statistical Methods in Medical Research*, 21(3):273–293, 2012. URL https://doi.org/10.1177/0962280210394483. [p293]

J. R. Zubizarreta. Stable weights that balance covariates for estimation with incomplete outcome data. *Journal of the American Statistical Association*, 110(511):910–922, 2015. URL https://doi.org/10.1080/01621459.2015.1023805. [p294]

Farhad Pishgar
Russell H. Morgan Department of Radiology and Radiological Science
Johns Hopkins University School of Medicine, Baltimore
United States
ORCiD: 0000-0003-0703-8442
Pishgar@JHMI.edu

Noah Greifer
Department of Mental Health
Johns Hopkins Bloomberg School of Public Health, Baltimore
United States
ORCiD: 0000-0003-3067-7154
NGreife1@JHU.edu

Clémence Leyrat
Department of Medical Statistics, Faculty of Epidemiology and Population Health
London School of Hygiene & Tropical Medicine, London
United Kingdom
ORCiD: 0000-0002-4097-4577
Clemence.Leyrat@lshtm.ac.uk

Elizabeth Stuart
Department of Mental Health & Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health, Baltimore
United States
ORCiD: 0000-0002-9042-8611
EStuart@JHU.edu

EMSS: New EM-type algorithms for the Heckman selection model in R

by Kexuan Yang, Sang Kyu Lee, Jun Zhao and Hyoung-Moon Kim

Abstract When investigators observe non-random samples from populations, sample selectivity problems may occur. The Heckman selection model is widely used to deal with selectivity problems. Based on the EM algorithm, Zhao et al. (2020) developed three algorithms, namely, ECM, ECM(NR), and ECME(NR), which also have the EM algorithm's main advantages: stability and ease of implementation. This paper provides the implementation of these three new EM-type algorithms in the package **EMSS** and illustrates the usage of the package on several simulated and real data examples. The comparison between the maximum likelihood estimation method (MLE) and three new EM-type algorithms in robustness issues is further discussed.

Introduction

The problem arising from the sampling mechanism where an investigator extracts a sample non-randomly, and then this sample cannot represent the population is usually referred to as a sample selection problem. Methods relying on a distributional assumption are widely used to deal with this selection problem. A classical sample selection model under the assumption of bivariate normality is introduced in Heckman (1974), and it is commonly called the Heckman selection model. Heckman (1979) further developed two estimation procedures for the above Heckman selection model: the maximum likelihood estimation method (MLE) and the two-step method.

The application of these two methods in the Heckman selection model is first described in the R package **sampleSelection** by Toomet and Henningsen (2008). For the observations where outlying ones are considered in the Heckman selection model, Zhelonkin et al. (2016) found that the unboundedness of the influence functions in the two-step method leads to an arbitrary bias. Zhelonkin et al. (2016) developed a robust two-stage method that performs more robustly than the two-step method, and the **ssmrob** package is available for robust estimation and inference for the selection model.

Little and Rubin (2002, pp. 322-323) applied an EM algorithm, which is numerically stable and easily implemented, to estimate the parameters of the Heckman selection model. However, it is limited to the cases in which the two vectors of the observed characteristics in the Heckman selection model are the same. Zhao et al. (2020) extended three new EM-type algorithms: expectation-conditional maximization (ECM), expectation-conditional maximization with Newton-Raphson method (ECM(NR)), and Expectation/Conditional Maximization Either (ECME) to more general cases. They also have the main advantages of the EM algorithm, namely stability and ease of implementation. However, section 6 in Zhao et al. (2020) suggests that the ECME algorithms require much more time than other two EM-type algorithms in the same real data analysis. In addition, there is still no R package available for these EM-type algorithms. Meng and Rubin (1993), Liu and Rubin (1994), and McLachlan and Krishnan (2008) are helpful to understand the procedures of the ECM and ECME algorithms.

This study developed the ECME algorithm by first applying the Newton–Raphson method to reduce the estimation time. Then, it is proposed to describe these new EM-type algorithms in R. In the next section, the Heckman selection model is described in brief, followed by new algorithms, namely ECM, ECM(NR), and ECME. Next, the usage of the **EMSS** package is presented through simulation and real data examples. Then, the robustness issue is further discussed for the MLE method and the new EM-type algorithms. Under the conditions where the robustness issue arises from the initial values, the **EMSS** package is preferable to the **sampleSelection**. Because of the unreasonable results in the MLE or the two-step method, the "NA" might occur in the standard errors in the **sampleSelection**. However, the standard errors can be calculated effectively in the **EMSS** package in almost all cases. Finally, we provide a summary of this study.

The **EMSS** package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=EMSS` and the GitHub at `https://github.com/SangkyuStat/EMSS`. R code for the examples demonstrated herein has been provided as supplementary material. The supplementary code has been tested with **EMSS** version 1.1.1, and results presented herein have been produced with this version.

Model and algorithms

Heckman selection model

Suppose that the regression model of the outcome variable of interest is

$$Y_{i1} = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \;\; i = 1, \ldots, N.$$

Due to selection mechanism,

$$Y_{i2} = \mathbf{w}_i^\top \gamma + \eta_i, \;\; i = 1, \ldots, N,$$

we observe only N_1 out of N observations y_{i1} for which $y_{i2} > 0$ such that

$$u_i = I(y_{i2} > 0).$$

$\mathbf{x}_i \in \mathbb{R}^p$ and $\mathbf{w}_i \in \mathbb{R}^q$ are observed characteristics. In addition, vectors $\boldsymbol{\beta} \in \mathbb{R}^p$, $\gamma \in \mathbb{R}^q$ are unknown parameters. Assume that the error terms ϵ_i and η_i follow bivariate normality as in Heckman (1974), that is,

$$\begin{pmatrix} \epsilon_i \\ \eta_i \end{pmatrix} \overset{\text{i.i.d.}}{\sim} N_2 \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma^2 & \rho\sigma \\ \rho\sigma & 1 \end{pmatrix} \right],$$

where "*i.i.d.*" means independent and identically distributed, then (Y_{i1}, Y_{i2}) also follow bivariate normal distribution.

EM-type algorithms

ECM and ECM(NR) algorithms

The ECM algorithm is discussed first. In the Heckman selection model, it is assumed that we observe the first N_1 out of N y_{i1} observations. The observed data are $\mathbf{y}_{obs} = (y_{11}, \cdots, y_{N_1,1})^\top$, and the missing data are $\mathbf{y}_{mis} = ((y_{N_1+1,1}, \cdots, y_{N,1}), \mathbf{y}_2^\top)^\top$. Applying the invariance property of MLEs, the parameter $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \gamma, \sigma, \rho\}$ can be transformed to $\boldsymbol{\theta}^* = \{\boldsymbol{\beta}, \gamma, \psi^*, \rho^*\}$ with $\psi = \sigma^2(1-\rho^2)$, $\psi^* = \log(\psi)$, $\rho^* = \rho\sigma$ and

$$\sigma^2 = \exp(\psi^*) + (\rho^*)^2 \text{ and } \rho = \frac{\rho^*}{\sqrt{\exp(\psi^*) + (\rho^*)^2}}.$$

The complete data log-likelihood is

$$\begin{aligned} l_c(\boldsymbol{\theta}^*|\mathbf{y}) &= -N\log(2\pi) - \frac{N}{2}\log(\psi) - \frac{1}{2\psi}\left\{\sum_{i=1}^{N}(y_{i1} - \mathbf{x}_i^\top\boldsymbol{\beta})^2\right\} \\ &- \frac{1}{2}\left(1 + \frac{(\rho^*)^2}{\psi}\right)\left\{\sum_{i=1}^{N}(y_{i2} - \mathbf{w}_i^\top\gamma)^2\right\} + \frac{\rho^*}{\psi}\left\{\sum_{i=1}^{N}(y_{i1} - \mathbf{x}_i^\top\boldsymbol{\beta})(y_{i2} - \mathbf{w}_i^\top\gamma)\right\}. \end{aligned}$$

The corresponding Q-function, which is the conditional expectation of the above complete-data log-likelihood $l_c(\boldsymbol{\theta}^*|\mathbf{y})$ with respect to the conditional distribution of $\mathbf{y}_{mis}$ given $\mathbf{y}_{obs}$ at the k-th iteration, is obtained as

$$\begin{aligned} Q\left(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)}\right) &= E\left[l_c(\boldsymbol{\theta}^*|\mathbf{y})|\hat{\boldsymbol{\theta}}^{*(k)}\right] = -N\log(2\pi) - \frac{N}{2}\log(\psi) \\ &- \frac{1}{2\psi}\left\{\sum_{i=1}^{N_1}(y_{i1} - \mathbf{x}_i^\top\boldsymbol{\beta})^2 + \sum_{i=N_1+1}^{N}(\hat{v}_{1m}^{(k)} - 2\hat{\alpha}_{1m}^{(k)}\mathbf{x}_i^\top\boldsymbol{\beta} + \boldsymbol{\beta}^\top\mathbf{x}_i\mathbf{x}_i^\top\boldsymbol{\beta})\right\} \\ &- \frac{1}{2}\left(1 + \frac{(\rho^*)^2}{\psi}\right)\left\{\sum_{i=1}^{N_1}(\hat{v}_{2o}^{(k)} - 2\hat{\alpha}_{2o}^{(k)}\mathbf{w}_i^\top\gamma + \gamma^\top\mathbf{w}_i\mathbf{w}_i^\top\gamma)\right. \\ &\qquad\qquad \left. + \sum_{i=N_1+1}^{N}(\hat{v}_{2m}^{(k)} - 2\hat{\alpha}_{2m}^{(k)}\mathbf{w}_i^\top\gamma + \gamma^\top\mathbf{w}_i\mathbf{w}_i^\top\gamma)\right\} \end{aligned}$$

$$+\frac{\rho^*}{\psi}\left\{\sum_{i=1}^{N_1}(y_{i1}\hat{\alpha}_{2o}^{(k)}-y_{i1}\mathbf{w}_i^\top\gamma-\hat{\alpha}_{2o}^{(k)}\mathbf{x}_i^\top\boldsymbol{\beta}+\boldsymbol{\beta}^\top\mathbf{x}_i\mathbf{w}_i^\top\gamma)\right.$$
$$\left.\sum_{i=N_1+1}^{N}(\hat{\alpha}_{12m}^{(k)}-\hat{\alpha}_{1m}^{(k)}\mathbf{w}_i^\top\gamma-\hat{\alpha}_{2m}^{(k)}\mathbf{x}_i^\top\boldsymbol{\beta}+\boldsymbol{\beta}^\top\mathbf{x}_i\mathbf{w}_i^\top\gamma)\right\},$$

where specific vectors are provided in the Appendix.

The ECM(NR) algorithm is developed based on the former ECM algorithm to accelerate the convergence process. The only difference is $\hat{\psi}^{*(k+1)}$ in the CM-steps (Conditional Maximization-steps) is first updated using the Newton–Raphson method, and $\hat{\psi}^{(k+1)}$ is then obtained. The specific expressions can be obtained in Zhao et al. (2020). For a better understanding of the ECM algorithm, Meng and Rubin (1993) and McLachlan and Krishnan (2008) can be referred to.

ECME algorithm

In this section, the ECME algorithm is briefly introduced and further developed to save running time. Liu and Rubin (1994) and McLachlan and Krishnan (2008) can be referred to in understanding the ECME algorithm in detail.

Assume that the complete data is $(\mathbf{z},\mathbf{y}_{obs},\mathbf{u})$, where $\mathbf{z}=(z_1,\cdots,z_{N_1})^\top$ is missing data, $\mathbf{y}_{obs}=(y_{11},\cdots,y_{N_1,1})^\top$, and $\mathbf{u}=(u_1,\cdots,u_N)^\top$. The related parameters $\boldsymbol{\theta}=\{\boldsymbol{\beta},\gamma,\sigma,\rho\}$ are transformed to $\boldsymbol{\theta}^*=\{\boldsymbol{\beta},\gamma,\psi^*,\rho^*\}$ like those in ECM algorithm. The complete data log-likelihood function can be written as

$$l_c(\boldsymbol{\theta}^*;\mathbf{z},\mathbf{y}_{obs},\mathbf{u})=-\frac{1}{2}\sum_{i=1}^{N}u_i\log(2\pi\psi)-\sum_{i=1}^{N}\frac{u_i\left(y_{i1}-\mathbf{x}_i^\top\boldsymbol{\beta}-\rho\sigma z_i\right)^2}{2\psi}$$
$$-\frac{1}{2}\sum_{i=1}^{N}u_i\log(2\pi)-\frac{1}{2}\sum_{i=1}^{N}u_iz_i^2+\sum_{i=1}^{N}(1-u_i)\log\left(\Phi(-\mathbf{w}_i^\top\gamma)\right),$$

and the following is the Q-function (which is the conditional expectation of the complete-data log-likelihood $l_c(\boldsymbol{\theta}^*;\mathbf{z},\mathbf{y}_{obs},\mathbf{u})$ with respect to the conditional distribution of $\mathbf{z}$ given $\mathbf{y}_{obs}$ and $\mathbf{u}$) calculated at the k-th iteration of the E-step:

$$Q(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)})=E\left[l_c(\boldsymbol{\theta}^*|\mathbf{z},\mathbf{y}_{obs},\mathbf{u})|\hat{\boldsymbol{\theta}}^{*(k)}\right]=-\frac{1}{2}\sum_{i=1}^{N}u_i\log(2\pi\psi)$$
$$-\frac{1}{2\psi}\sum_{i=1}^{N}\left\{u_i(y_{i1}-\mathbf{x}_i^\top\boldsymbol{\beta})^2-2u_i(y_{i1}-\mathbf{x}_i^\top\boldsymbol{\beta})\rho^*\hat{\alpha}_i^{(k)}+u_i(\rho^*)^2\hat{\delta}_i^{(k)}\right\}$$
$$-\frac{1}{2}\sum_{i=1}^{N}u_i\log(2\pi)-\frac{1}{2}\sum_{i=1}^{N}u_i\hat{\delta}_i^{(k)}+\sum_{i=1}^{N}(1-u_i)\log(\Phi(-\mathbf{w}_i^\top\gamma)),$$

where

$$\hat{\alpha}_i^{(k)}=E[Z_i|\hat{\theta}^{*^{(k)}},y_{i1},U_i=1]\text{ and}$$
$$\hat{\delta}_i^{(k)}=E[Z_i^2|\hat{\theta}^{*^{(k)}},y_{i1},U_i=1],$$

with the conditional distribution

$$Z_i|\hat{\theta}^{*^{(k)}},y_{i1},U_i=1\sim TN_{(-\mathbf{w}_i^\top\hat{\gamma}^{(k)},\infty)}\left(\frac{\hat{\rho}^{(k)}}{\hat{\sigma}^{(k)}}\left(y_{i1}-\mathbf{x}_i^\top\hat{\boldsymbol{\beta}}^{(k)}\right),1-\hat{\rho}^{2^{(k)}}\right),$$

where the stochastic representation of the density function for $y_{i1}|u_i=1$ is considered. The ECME algorithm is time-consuming because calculating $\hat{\gamma}^{(k+1)}$ requires a significant amount of time. The Newton-Raphson method is applied to reduce the computing time. In the CM-step, the $\hat{\gamma}^{(k)}$ is updated by

$$\hat{\gamma}^{(k+1)}=\hat{\gamma}^{(k)}-\left[\frac{\partial^2}{\partial\gamma\partial\gamma^\top}\log L\left(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)}\right)\right]^{-1}\frac{\partial}{\partial\gamma}\log L\left(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)}\right),$$

where

$$\frac{\partial}{\partial\gamma}\log L\left(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)}\right)=\sum_{i=1}^{N}u_i\frac{\phi(A)}{\Phi(A)}\frac{\hat{\sigma}^{(k+1)}}{\sqrt{\exp\left(\hat{\psi}^{*(k+1)}\right)}}\mathbf{w}_i+\sum_{i=1}^{N}(u_i-1)\frac{\phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}{\Phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}\mathbf{w}_i,$$

and

$$\begin{aligned}\frac{\partial^2\log L\left(\boldsymbol{\theta}^*|\hat{\boldsymbol{\theta}}^{*(k)}\right)}{\partial\gamma\partial\gamma^\top}=&-\sum_{i=1}^{N}\frac{u_i\hat{\sigma}^{(k+1)^2}}{\exp\left(\hat{\psi}^{*(k+1)}\right)}\left[A\frac{\phi(A)}{\Phi(A)}+\left(\frac{\phi(A)}{\Phi(A)}\right)^2\right]\mathbf{w}_i\mathbf{w}_i^\top\\&+\sum_{i=1}^{N}(1-u_i)\left[\frac{\mathbf{w}_i^\top\gamma\phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}{\Phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}-\left(\frac{\phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}{\Phi\left(-\mathbf{w}_i^\top\gamma^{(k+1)}\right)}\right)^2\right]\mathbf{w}_i\mathbf{w}_i^\top,\end{aligned}$$

with

$$A=\frac{\hat{\sigma}^{(k+1)}\mathbf{w}_i^\top\hat{\gamma}^{(k)}+\hat{\rho}^{(k+1)}\left(y_{i1}-\mathbf{x}_i^\top\hat{\boldsymbol{\beta}}^{(k+1)}\right)}{\sqrt{\exp\left(\hat{\psi}^{*(k+1)}\right)}}.$$

Implementation in EMSS

The package **EMSS** is constructed to describe the three EM-type algorithms. In **EMSS**, the main function for the estimation of the Heckman selection model is EMSS. A formula for the response equation whose argument is response and a formula for the selection equation with argument selection are required. With the default estimation method ECM ("ECM"), the user can also choose the method "ECMnr" for the ECM(NR) method or "EMCE" for the ECME method. The argument initial.param can be used to set the initial values. If the initial values are not provided by the user, **EMSS** conducts the estimation of the consistent initial values offered by the two-step method through the package **sampleSelection**.

The result of **EMSS** is a list of class 'EMSS', and several methods for the objects of this class are also provided by the package **EMSS**. Command print prints the estimation results. Command summary calculates and prints the summarized results. coef extracts the estimated coefficients, and vcov extracts the variance-covariance matrix. confint can be used to calculate the confidence intervals of all the parameters by applying the following equation.

$$\left[\ \hat{\text{para}}+Z_\alpha\times\text{stdrr}\quad \hat{\text{para}}+Z_{1-\alpha}\times\text{stdrr}\ \right],$$

where $\hat{\text{para}}$ is the estimated value of a parameter, stdrr is the corresponding standard error value, and Z_α and $Z_{1-\alpha}$ are the quantile values of standard normal distribution at α and $1-\alpha$, respectively, with $\alpha=(1-\text{level})/2$, where "level" is the confidence level. The default confidence level (level) is 0.95 (95%), and it can be changed to any value between 0 and 1.

Using EMSS

This section illustrates the usage of **EMSS** using a simulation example and application to a real data set. An example using random numbers is given first, with exclusion restriction where the two observed characters **X** and **W** are not the same.

```
set.seed(0)
library( mvtnorm )
N<-1000
errps<-rmvnorm(N,c(0,0),matrix(c(1,0.5,0.5,1),2,2) )
w<-runif(N)
y2<-w+errps[,1]>0
x<-runif(N)
y1<-(x+errps[,2])*(w>0)
```

The package mvtnorm is used to create bivariate normal disturbances with a correlation of 0.5. The observed character for selection, w, is generated by uniform distribution, and the selection outcome y2 is then generated using the probit generating process. Through a similar process, the explanatory variable x and the outcome variable of interest y1 are generated. Note that the two observed characters, w and x, are independent and thus fulfill the exclusion restriction. Hence, the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are set equally as $(0,1)^\top$ and must be estimated. The estimated results in the ECM algorithm are as follows.

```
summary(EMSS(response=y1~x,selection=y2~w))

Call:
EMSS(response = y1 ~ x, selection = y2 ~ w)

Q-Value: -2637.487

Response equation:
             Estimate Std. Error Z Value  Pr(>|Z|)
(Intercept)  -0.2904     0.1258  -2.309 2.096e-02 *
x             1.2319     0.1311   9.398 5.548e-21 ***

Selection equation:
              Estimate Std. Error Z Value  Pr(>|Z|)
(Intercept)   0.1010    0.07628   1.324 1.855e-01
w             0.7569    0.13243   5.716 1.093e-08 ***
---

Sigma:
         Estimate Std. Error Z Value  Pr(>|Z|)
sigma    1.124    0.07167   15.69 1.797e-55 ***

Rho:
      Estimate Std. Error Z Value  Pr(>|Z|)
rho   0.6858     0.1214    5.65 1.603e-08 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated results for the parameters are reasonably precise.

The following real data example is an example in Cameron and Trivedi (2009, Section 16.6.5, p. 546) regarding ambulatory expenditures from the 2001 Medical Expenditure Panel Survey. The data consist of 3328 observations with 526 corresponding to zero expenditures and is available in MEPS2001 of the R package **ssmrob**. To estimate an individual's medical expenditures, the outcome (response) variable of interest, log ambulatory expenditures (lnambx), is modeled by individual's age (age), gender (female), education attainment in years (educ), ethnicity (blhisp), number of chronic diseases (totchr), and insurance status (ins). The selection variable, ambulatory expenditures, which is described by dambexp is modeled by all the former regressors and the income variable (income). The model is estimated using the ECM(NR) method.

```
library(ssmrob)
data(MEPS2001))
outcomeEq<-lnambx ~ age+female+educ+blhisp+totchr+ins
selectEq<-dambexp ~ age+female+educ+blhisp+totchr+ins+income
summary(EMSS(response=outcomeEq, selection=selectEq,
                data=MEPS2001,method="ECMnr"))


Call:
EMSS(response = outcomeEq, selection = selectEq, data = MEPS2001,
method = "ECMnr")

Q-Value: -10213.94

Response equation:
             Estimate  Std. Error Z Value   Pr(>|Z|)
(Intercept)  5.04406    0.22813  22.111  2.493e-108 ***
age          0.21197    0.02301   9.213   3.160e-20 ***
femaleTRUE   0.34814    0.06011   5.791   6.984e-09 ***
educ         0.01872    0.01055   1.774   7.599e-02 .
blhispTRUE  -0.21857    0.05967  -3.663   2.492e-04 ***
```

```
totchr       0.53992    0.03933  13.727  6.996e-43 ***
insTRUE     -0.02999    0.05109  -0.587  5.572e-01

Selection equation:
            Estimate   Std. Error Z Value  Pr(>|Z|)
(Intercept) -0.676054   0.194029  -3.484 4.934e-04 ***
age          0.087936   0.027421   3.207 1.342e-03 **
femaleTRUE   0.662665   0.060938  10.874 1.528e-27 ***
educ         0.061948   0.012029   5.150 2.609e-07 ***
blhispTRUE  -0.363938   0.061873  -5.882 4.054e-09 ***
totchr       0.796951   0.071131  11.204 3.895e-29 ***
insTRUE      0.170137   0.062871   2.706 6.807e-03 **
income       0.002708   0.001317   2.056 3.975e-02 *
---

Sigma:
         Estimate Std. Error Z Value Pr(>|Z|)
sigma    1.271    0.01838   69.16        0 ***

Rho:
     Estimate Std. Error Z Value Pr(>|Z|)
rho  -0.1306     0.1471  -0.888   0.3746
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All coefficients and standard errors are completely identical to the results reported in Cameron and Trivedi (2009, Section 16.6.5, p. 546).

The confidence interval of this example can be calculated using the following codes.

```
 confint(EMSS(response=outcomeEq, selection=selectEq,
              data=MEPS2001,method="ECMnr"))
```

To compare the computing times of the original ECME algorithm and the developed one for which the Newton-Raphson method is applied, the real data example in Section 6, Zhao et al. (2020), is analyzed again. The developed ECME algorithm is used as follows.

```
library(sampleSelection)
data(Mroz87)
selectEq <- lfp ~ age + I(age^2)+ faminc + kids5 + educ
outcomeEq <- wage ~ exper + I(exper^2) + educ + city
EMSS(response = outcomeEq, selection = selectEq, data = Mroz87,
        method = "ECME")
```

The results are similar to those of the ECM and ECM(NR) algorithm in table 8 of Zhao et al. (2020), which are slightly better than those in the original ECME algorithm.

The **EMSS** package and R codes were executed on a computer with an Intel(R) Core (TM) i7-4790M CPU at 3.60 GHz, running MS-Windows 10. The ECME algorithm developed herein takes 22.67 s while the original one takes 14.63 min. The computing time of the ECME algorithm is thus significantly reduced than before.

Robustness issues

Zhao et al. (2020) concluded that a robustness issue arises from the initial values in the MLE method but not in the three EM-type algorithms. Here, we aim to discuss this robustness issue further. For the simulated example in Section Using **EMSS**, the MLE method is also applied to estimate the data set using the command `selection()` in R package **sampleSelection** (Henningsen et al., 2019). The initial values are found to influence the estimated values of parameters in the MLE method. For example, if the initial value of σ is set to 5, that of ρ is set to 0.8, and those of other parameters are set to 0s, the results of the MLE method are given as follows.

```
summary(selection(y2~w,y1~x,start=c(rep(0,4), 5,0.8) ),method="ml")

--------------------------------------------
Tobit 2 model (sample selection model)
Maximum Likelihood estimation
```

```
Newton-Raphson maximization, 3 iterations
Return code 3: Last step could not find a value above the current.
Boundary of parameter space?
Consider switching to a more robust optimization method temporarily.
Log-Likelihood: -2214.037
1000 observations (318 censored and 682 observed)
6 free parameters (df = 994)
Probit selection equation:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.05991    0.03482   1.721   0.0856 .
w            0.03362    0.05204   0.646   0.5185
Outcome equation:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02336    0.05811  -0.402    0.688
x           -0.01041    0.22132  -0.047    0.963
Error terms:
        Estimate Std. Error t value Pr(>|t|)
sigma   5.0000         NA      NA       NA
rho     0.9652         NA      NA       NA
--------------------------------------------
Warning messages:
1: In sqrt(diag(vc)) : NaNs produced
2: In sqrt(diag(vc)) : NaNs produced
3: In sqrt(diag(vcov(object, part = "full"))) : NaNs produced
```

The p-values of the estimated parameters (except σ and ρ) suggest that the estimated results are not significant. The occurrence of "NA" resulting from the variance-covariance matrix further implies that the estimated results in the MLE method are not reasonable. The previous section, Using **EMSS**, presents the results from the ECM algorithm where the initial values are set by default based on the results from the two-step method. With the different initial value sets in this section, the ECM, ECM(NR), and ECME algorithms perform stably, and their results are the same as the ones of the ECM algorithm shown in the Using **EMSS** section to eight decimal places. If the simulated annealing maximizer is applied to select the initial values for the MLE methods, the estimated results are similar to those in the three EM-type algorithms.

To avoid the occasion of the robustness issue in the MLE method, the former scenario is regenerated 1000 times with the degree of censoring corresponding to approximately 30%. Considering the same initial value set as the former, the boxplots in Figures 1 and 2 suggest that the three EM-type algorithms perform similarly and even much better than the MLE algorithm.

In Cameron and Trivedi (2005), the data set `RanHIE`, which is available in package **sampleSelection**, based on the "RAND Health Insurance Experiment" is used to analyze how the patient's use of health services is affected by the types of randomly assigned health insurance. An example based on the analysis in Cameron and Trivedi (2005, p. 553) is provided to further discuss this situation.

The outcome variable $\mathbf{y}_1$ is `lnmeddol`, which measures the log of an individual's medical expenses, and the selection variable $\mathbf{y}_2$ is `binexp`, which indicates whether the medical expenses are positive. The observed character $\mathbf{X}$ consists of the log of the coinsurance rate plus 1 (`logc=log(coins+1)`), the dummy for the individual deductible plan (`idp`), the log of participation incentive payment (`lpi`), the number of chronic diseases (`disea`), the log of family size (`lfam`), education of household head in years (`educdec`), age of individual in years (`xage`), quadratic polynomial in the age of individual in years, and a dummy variable for female individuals (`female`). The observed character $\mathbf{W}$ consists of `logc`, physical limitations (`physlm`), `disea`, quadratic polynomial in `disea`, `lfam`, `educdec`, `xage`, and `female`. A partial sample where the study year (`year`) is equal to 2 and the education information is given is selected for the estimation with sample size $N = 5,574$.

Fix the initial values of all parameters except σ and ρ at 0 and consider three different initial values sets for σ and ρ. If the initial values of σ range from 0.1 to 0.65, then the estimated values of all parameters in the MLE method are the corresponding initial values regardless of the initial values of ρ. The estimated results in the ECM algorithm are stable as follows.

```
data(RandHIE)
subsample<-RandHIE$year==2&!is.na(RandHIE$educdec)
outcomeEq<-lnmeddol~logc+physlm+disea+I(disea^2)+lfam+educdec+xage+female
selectEq<-binexp~logc+idp+lpi+disea+lfam+educdec+xage+I(xage^2)+female
summary(EMSS(response=outcomeEq,selection=selectEq,
    initial.para=c(rep(0,19),0.2,0.5), data=RandHIE[subsample,]))
```

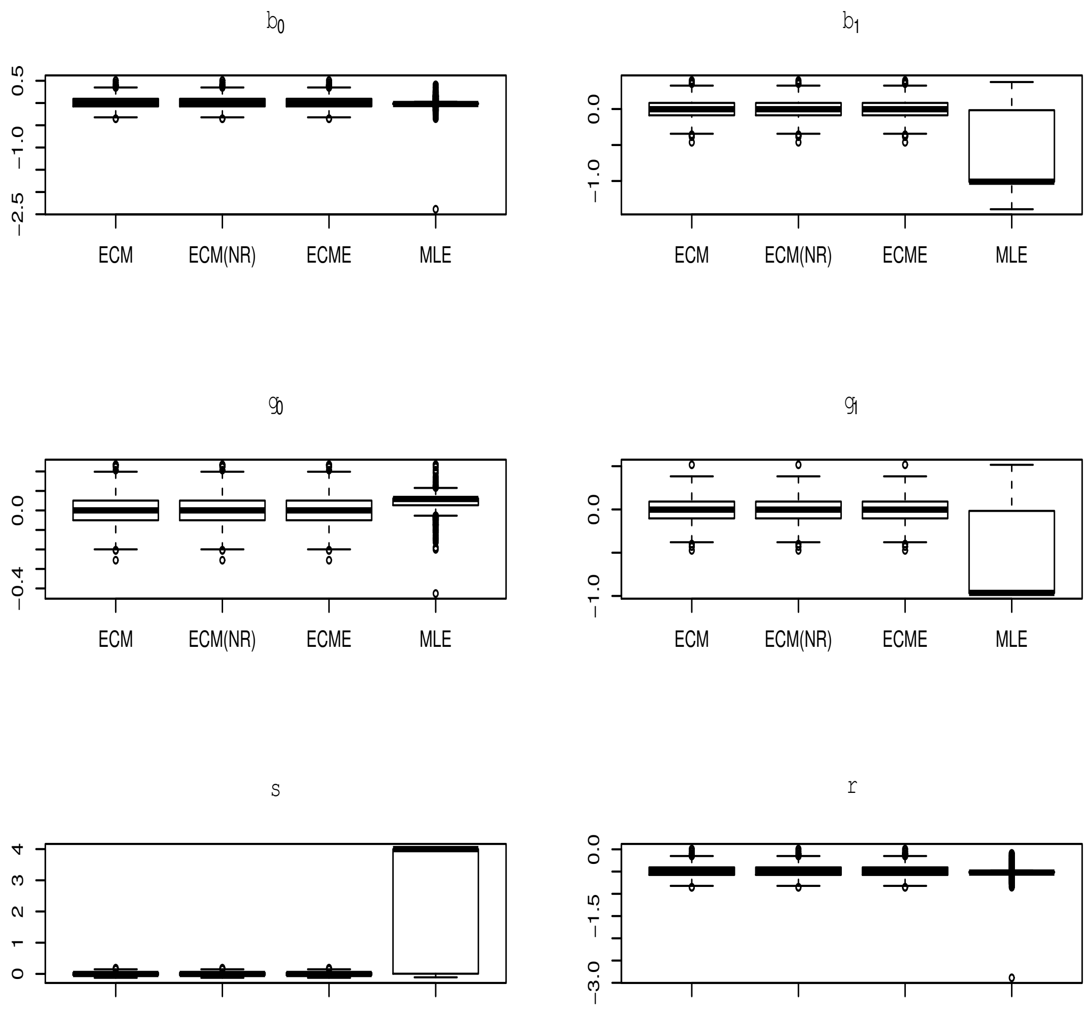


Figure 1: Boxplots of the bias in the estimation for the 1000 regeneration of the simulated example in the section Using **EMSS**.

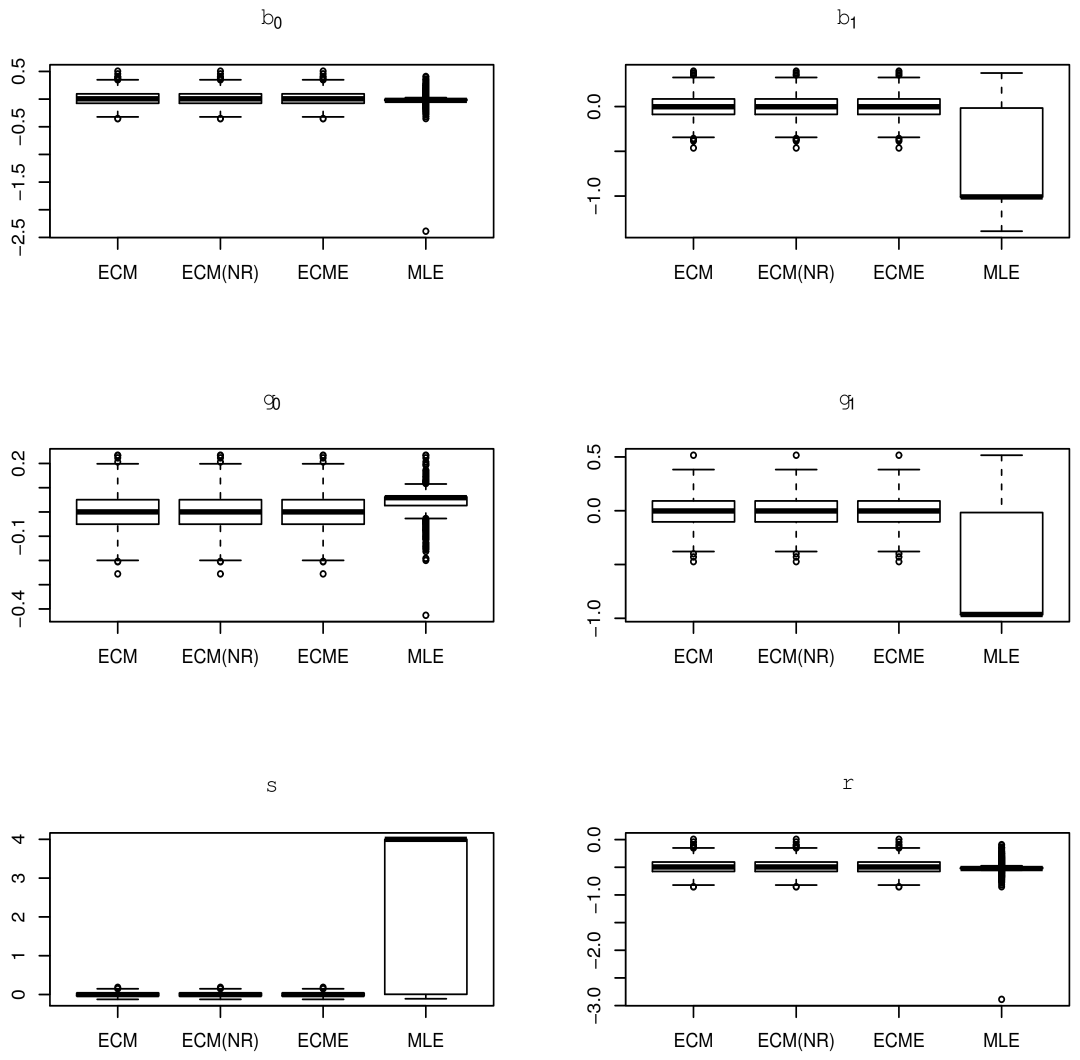


Figure 2: Boxplots of the MSE in the estimation for the 1000 regeneration of the simulated example in the section Using **EMSS**.

```
Call:
EMSS(response = outcomeEq, selection = selectEq,
data = RandHIE[subsample, ])

Q-Value: -16195.75

Response equation:
             Estimate   Std. Error  Z Value  Pr(>|Z|)
(Intercept)  2.4841461   0.168714  14.7240 4.523e-49 ***
logc        -0.1199851   0.011946 -10.0440 9.762e-24 ***
physlm       0.2952680   0.068552   4.3072 1.653e-05 ***
disea        0.0415756   0.008574   4.8490 1.241e-06 ***
I(disea^2)  -0.0001355   0.000250  -0.5421 5.878e-01
lfam        -0.1828111   0.048101  -3.8006 1.443e-04 ***
educdec      0.0350172   0.008674   4.0368 5.418e-05 ***
xage         0.0203750   0.001588  12.8310 1.100e-37 ***
female       0.3123718   0.048632   6.4231 1.335e-10 ***

Selection equation:
             Estimate   Std. Error  Z Value  Pr(>|Z|)
(Intercept) -0.0807292  1.240e-01  -0.651 5.150e-01
logc        -0.1138537  1.078e-02 -10.562 4.470e-26 ***
idp         -0.0632783  3.994e-02  -1.584 1.131e-01
lpi          0.0320468  7.301e-03   4.389 1.136e-05 ***
disea        0.0283038  3.329e-03   8.503 1.841e-17 ***
lfam        -0.0666747  3.799e-02  -1.755 7.922e-02 .
educdec      0.0516196  6.923e-03   7.456 8.928e-14 ***
xage        -0.0051879  4.066e-03  -1.276 2.020e-01
I(xage^2)    0.0001979  6.907e-05   2.865 4.164e-03 **
female       0.2098103  3.829e-02   5.479 4.281e-08 ***
---

Sigma:
         Estimate Std. Error Z Value Pr(>|Z|)
sigma    1.604    0.02888   55.56        0 ***

Rho:
      Estimate Std. Error Z Value   Pr(>|Z|)
rho    0.745     0.0323   23.07   9.724e-118 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the second initial value set, the initial value of ρ is 0.5, and that of σ is within $[0.66, 7.9]$. The three new EM-type algorithms perform similarly to the ECM algorithm in the first initial value set. The estimated results in the MLE method are stable like those in the ECM algorithm.

In the third initial value set, the initial values of σ are larger than 7.9, and the three new EM-type algorithms still perform stably, similar to the former. However, the estimated values of each parameter in the MLE method are not stable. For example,

```
summary(selection(selectEq,outcomeEq,start=c(rep(0,19),8.8,0.5),
           data=RandHIE[subsample,],method="ml" ))

--------------------------------------------
Tobit 2 model (sample selection model)
Maximum Likelihood estimation
Newton-Raphson maximization, 3 iterations
Return code 3: Last step could not find a value above the current.
Boundary of parameter space?
Consider switching to a more robust optimization method temporarily.
Log-Likelihood: -15707.81
5574 observations (1293 censored and 4281 observed)
21 free parameters (df = 5553)
Probit selection equation:
             Estimate   Std. Error  t value Pr(>|t|)
(Intercept)  2.155e-03  1.029e-01   0.021   0.9833
```

```
logc         5.082e-03  8.366e-03   0.607   0.5436
idp          1.106e-03  2.536e-02   0.044   0.9652
lpi          6.242e-04  4.436e-03   0.141   0.8881
disea       -4.776e-03  2.504e-03  -1.907   0.0565 .
lfam         4.923e-03  3.258e-02   0.151   0.8799
educdec     -1.921e-03  5.710e-03  -0.336   0.7366
xage        -2.601e-03  2.440e-03  -1.066   0.2866
I(xage^2)   -1.355e-05  3.918e-05  -0.346   0.7294
female      -2.802e-03  3.258e-02  -0.086   0.9315
Outcome equation:
             Estimate   Std. Error  t value Pr(>|t|)
(Intercept) -0.0056280  0.8243005  -0.007  0.99455
logc        -0.0120012  0.0616586  -0.195  0.84568
physlm       0.0001365  0.2694167   0.001  0.99960
disea       -0.0359361  0.0349727  -1.028  0.30421
I(disea^2)   0.0001022  0.0009454   0.108  0.91390
lfam        -0.0086017  0.2567682  -0.033  0.97328
educdec     -0.0622248  0.0449205  -1.385  0.16604
xage        -0.0286630  0.0081770  -3.505  0.00046 ***
female      -0.0023083  0.2580752  -0.009  0.99286
Error terms:
        Estimate Std. Error t value Pr(>|t|)
sigma   8.8015         NA      NA       NA
rho     0.9186         NA      NA       NA
--------------------------------------------
Warning messages:
1: In sqrt(diag(vc)) : NaNs produced
2: In sqrt(diag(vc)) : NaNs produced
3: In sqrt(diag(vcov(object, part = "full"))) : NaNs produced
```

The warning messages, the p-value of each parameter, and the "NA" in the standard errors of parameters σ, and ρ suggest that the estimated results of the MLE method are not reasonable.

By summarizing the above three initial value sets with setting the initial values of σ at $\{0.1, 0.3, \ldots, 10\}$, Figure 3 presents the histograms of the estimated coefficients for the variables physlm, female, and the estimated values of the parameter σ in all four algorithms. The histograms illustrate that the horizontal axis value in the three EM-type algorithms are the same all the time, which further suggests that three EM-type algorithms do not affect by the initial values, and they perform similarly. However, the histograms of the MLE method imply that the MLE method is not stable.

This suggests that the three EM-type algorithms are more robust than the MLE method. However, the computing times of the above three examples using the ECME algorithm are relatively longer than those of ECM and ECM(NR). It is found that the computing time of the ECME algorithm is affected by the sample size. For instance, under different sample sizes, Table 1 presents the running time of the ECM, ECM(NR), and ECME algorithms for the simulated example shown in the section using **EMSS** with 10-times regeneration. As the sample size increases, the ECME algorithm costs much more time than the ECM and ECM(NR) algorithms.

Sample sizes n	200	300	500	800	1000
ECM	2.5415	3.1542	4.4769	6.1861	7.8049
ECM(NR)	2.5232	3.0921	4.4393	6.0212	7.6943
ECME	7.7486	16.9576	26.7005	119.0904	159.1983

Table 1: Running time (seconds) of the simulated example in the section Using **EMSS** with 10-times regeneration.

If the sample size of the former RandHIE data example decreases to $1,000$ randomly with the same outcome and selection models, MLE presents results similar to the three EM-type algorithms only when the initial values of σ ranges from 0.71 to 7.0 (the values may change since the sample size is randomly reduced to $1,000$). The three EM-type algorithms still present stable results that are similar to each other. Furthermore, the EMCE algorithm takes a similar computing time with the ECM and ECM(NR). So it will be better to use ECM or ECM(NR) algorithms for the large-size samples.

To achieve more robust estimation, the simulated annealing maximizer for 10,000 iterations is applied to offer better initial values for the MLE method. Note that the selected value for "parscale" in

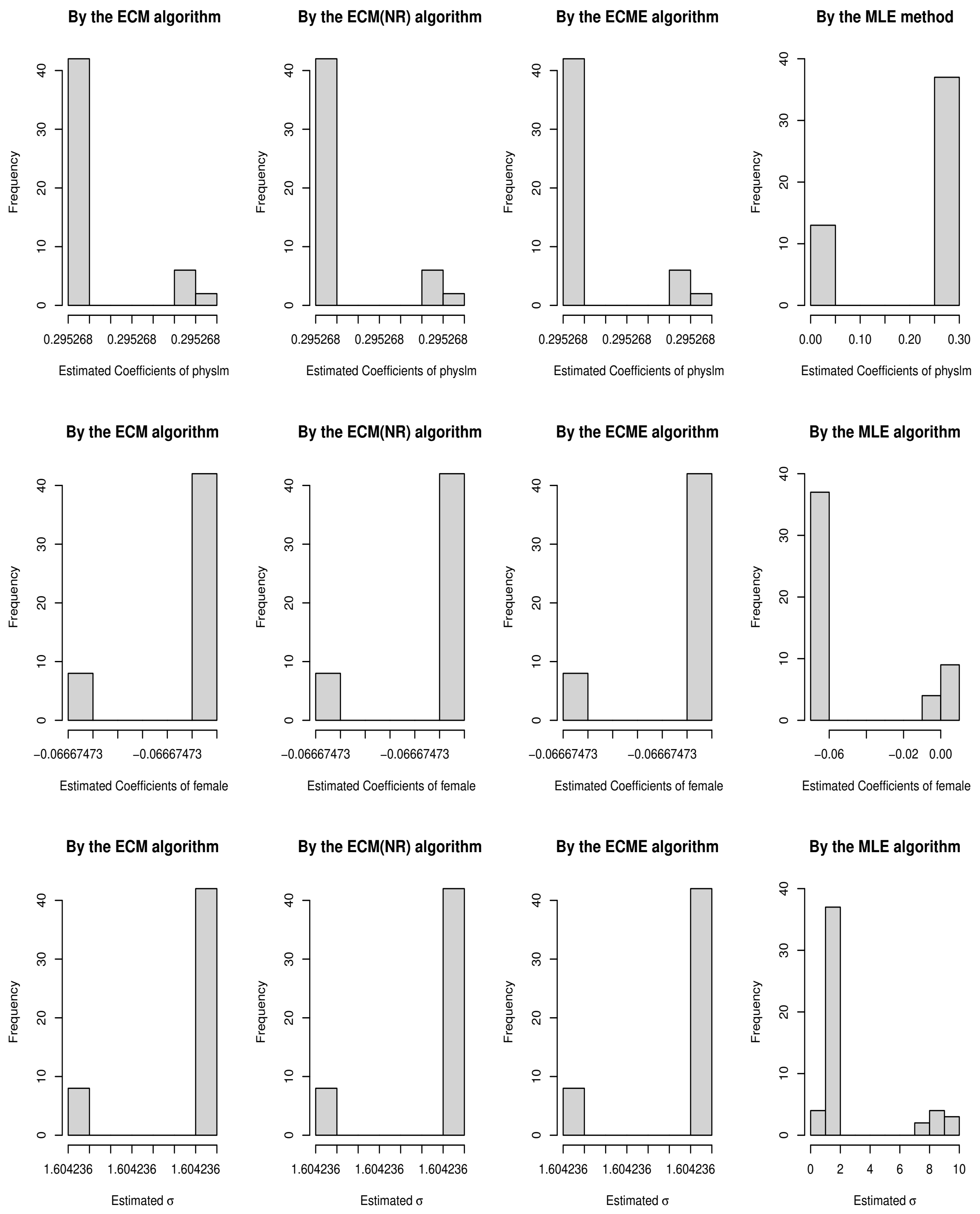


Figure 3: Some estimated values in the four algorithms for the `RanHIE` data.

SANN is 0.001, which was satisfactory for this data set. The estimated results of the parameters are the same as those in the MLE method in the second initial value set. The calculated log-likelihood value is -10331.12, which is greater than -15707.81 in the third initial value set.

Discussion

The ECME algorithm is developed through the application of the Newton-Raphson method to reduce the computing time. The implementation of three new EM-type algorithms, namely ECM, ECM(NR), and ECME, are described in package **EMSS**. The application of the package **EMSS** is conducted using simulated and real data sets. The examples for which initial values are considered in detail further confirm that the three new EM-type algorithms are more robust than the MLE method. The **EMSS** package is preferable to the **sampleSelection** when the robustness issue arising from the initial values is involved. The standard errors might not be calculated appropriately in the MLE or the two-step method in the **sampleSelection** because of the unreasonable results, but they can always be calculated effectively using the **EMSS** package.

Acknowledgment

The corresponding author's research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07045603) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A4A5032622).

Bibliography

A. C. Cameron and P. K. Trivedi. *Microeconometrics: Methods and Applications*. Cambridge University Press, New York, 2005. [p312]

A. C. Cameron and P. K. Trivedi. *Microeconometrics using Stata*. Stata Press, College Station, Texas, 2009. [p310, 311]

J. Heckman. Shadow prices, market wages, and labor supply. *Econometrica*, 42(4):679–694, 1974. URL http://www.jstor.org/stable/1913937. [p306, 307]

J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–161, 1979. URL https://www.jstor.org/stable/1912352. [p306]

A. Henningsen, O. Toomet, and S. Petersen. **sampleSelection***: Sample Selection Models*. R Foundation for Statistical Computing, 2019. URL https://cran.r-project.org/web/packages/sampleSelection/index.html. [p311]

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley & Sons, Inc., New Jersey, 2002. doi: 10.1002/9781119013563. [p306]

C. Liu and D. Rubin. The ecme algorithm: A simple extension of em and ecm with faster monotone convergence. *Biometrika*, 81:633—-648, 1994. doi: 10.1093/biomet/81.4.633. [p306, 308]

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions, second ed*. Wiley, New Jersey, 2008. [p306, 308]

X. Meng and D. Rubin. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80:267—-278, 1993. doi: 10.1093/biomet/80.2.267. [p306, 308]

O. Toomet and A. Henningsen. Sample selection models in r: Package sampleselection. *Econometrica*, 27(7):1–23, 2008. doi: 10.18637/jss.v027.i07. [p306]

J. Zhao, H.-J. Kim, and H.-M. Kim. New em-type algorithms for the heckman selection model. *Computational Statistics and Data Analysis*, 146:waiting, 2020. doi: 10.1016/j.csda.2020.106930. [p306, 308, 311]

M. Zhelonkin, M. G. Genton, and E. Ronchetti. Robust inference in sample selection models. *Journal of the Royal Statistical Society, Series B*, 78:805—827, 2016. doi: 10.1111/rssb.12136. [p306]

Kexuan Yang
Department of Applied Statistics, Konkuk University
120 Neungdong-ro, Gwangjin-gu, Seoul
South Korea
717260446@qq.com

Sang Kyu Lee
Department of Statistics and Probability, Michigan State University
619 Red Cedar Road East Lansing, MI 48824-1027
the USA
lsk0816@gmail.com

Jun Zhao

School of Mathematics and Statistics, Ningbo University
No. 818, Fenghua Road, Ningbo, Zhejiang
China
zhaojun2021@hotmail.com

Hyoung-Moon Kim

Department of Applied Statistics, Konkuk University
120 Neungdong-ro, Gwangjin-gu, Seoul
South Korea
hmkim@konkuk.ac.kr

Appendix

Some vectors in the Q-function of ECM and ECM(NR) algorithms at the k-th iteration are presented herewith. For missing y_{i1},

i) $\hat{\alpha}_{1m}^{(k)} = E(Y_{i1}|Y_{i2} \leq 0) = \hat{\mu}_{i1}^{(k)} - \hat{\rho^*}^{(k)}\lambda(-\hat{\mu}_{i2}^{(k)})$, where $\hat{\mu}_{i1}^{(k)} = \mathbf{x}_i^\top \hat{\boldsymbol{\beta}}^{(k)}$ and $\hat{\mu}_{i2}^{(k)} = \mathbf{w}_i^\top \hat{\boldsymbol{\gamma}}^{(k)}$;

ii) $\hat{\alpha}_{2m}^{(k)} = E(Y_{i2}|Y_{i2} \leq 0) = \hat{\mu}_{i2}^{(k)} - \lambda(-\hat{\mu}_{i2}^{(k)})$;

iii) $\hat{\nu}_{1m}^{(k)} = E(Y_{i1}^2|Y_{i2} \leq 0) = \hat{\mu}_{i1}^{2(k)} + \hat{\sigma}^{2(k)} - \hat{\rho^*}^{(k)}\lambda(-\hat{\mu}_{i2}^{(k)})(2\hat{\mu}_{i1}^{(k)} - \hat{\rho^*}^{(k)}\hat{\mu}_{i2}^{(k)})$;

iv) $\hat{\nu}_{2m}^{(k)} = E(Y_{i2}^2|Y_{i2} \leq 0) = 1 + \hat{\mu}_{i2}^{2(k)} - \hat{\mu}_{i2}^{(k)}\lambda(-\hat{\mu}_{i2}^{(k)})$;

v) $\hat{\alpha}_{12m}^{(k)} = E(Y_{i1}Y_{i2}|Y_{i2} \leq 0) = \hat{\mu}_{i1}^{(k)}(\hat{\mu}_{i2}^{(k)} - \lambda(-\hat{\mu}_{i2}^{(k)})) + \hat{\rho^*}^{(k)}$,

and for observed y_{i1},

vi) $\hat{\alpha}_{2o}^{(k)} = E(Y_{i2}|Y_{i1}, Y_{i2} > 0) = \hat{\mu}_{i2.1}^{(k)} + \sqrt{1 - \hat{\rho}^{2(k)}}\, \lambda\left(\dfrac{\hat{\mu}_{i2.1}^{(k)}}{\sqrt{1 - \hat{\rho}^{2(k)}}}\right)$;

$\hat{\nu}_{2o}^{(k)} = E(Y_{i2}^2|Y_{i1}, Y_{i2} > 0) = 1 - \hat{\rho}^{2(k)} + \hat{\mu}_{i2.1}^{2(k)} + \hat{\mu}_{i2.1}^{(k)}\sqrt{1 - \hat{\rho}^{2(k)}}\lambda\left(\dfrac{\hat{\mu}_{i2.1}^{(k)}}{\sqrt{1 - \hat{\rho}^{2(k)}}}\right)$, where $\hat{\mu}_{i2.1}^{(k)} =$
$\mathbf{w}_i^\top \boldsymbol{\gamma}^{(k)} + \dfrac{\hat{\rho}^{(k)}}{\hat{\sigma}^{(k)}}(y_{i1} - \mathbf{x}_i^\top \boldsymbol{\beta}^{(k)})$.

Analysis of Corneal Data in R with the rPACI Package

by Darío Ramos-López and Ana D. Maldonado

Abstract In ophthalmology, the early detection of keratoconus is still a crucial problem. Placido disk corneal topographers are essential in clinical practice, and many indices for diagnosing corneal irregularities exist. The main goal of this work is to present the R package **rPACI**, providing several functions to handle and analyze corneal data. This package implements primary indices of corneal irregularity (based on geometrical properties) and compound indices built from the primary ones, either using a generalized linear model or as a Bayesian classifier using a hybrid Bayesian network and performing approximate inference. **rPACI** aims to make the analysis of corneal data accessible for practitioners and researchers in the field. Moreover, a **shiny** app was developed to use **rPACI** in any web browser in a truly user-friendly graphical interface without installing R or writing any R code. It is openly deployed at `https://admaldonado.shinyapps.io/rPACI/`.

Introduction

One of the major problems in the field of ophthalmology is the early detection of keratoconus (Martínez-Abad and Piñero, 2017; Accardo and Pensiero, 2002; Rabinowitz, 1998). Keratoconus is a serious disease that deforms and weakens the cornea (the outer part of the eye). There are several grades of this disease (Alió and Shabayek, 2006), with the most advanced stages being easy to detect. However, early or subclinical keratoconus cases are much more difficult to identify (de-Sanctis et al., 2008; Maguire and Bourne, 1989). Diagnosing keratoconus is crucial before undergoing any kind of refractive eye surgery (Llorente et al., 2004), in which vision is corrected by modifying the corneal curvature, normally with lasers. Failure to detect early keratoconus in these circumstances can result in an ectasia (Randleman et al., 2003), which in the most severe cases can lead to blindness or corneal transplantation.

For the measurement and analysis of the cornea, the main clinical tool is the corneal topography (Fan et al., 2018; Piñero, 2015; Read et al., 2009; Rowsey et al., 1981). Depending on the employed technology, they can be basically classified into two types: Scheimpflug camera devices (Read et al., 2009) or Placido disk devices (Alkhaldi et al., 2009; Samapunphong and Azar, 1998; Rand et al., 1997), although some devices combine these two technologies (Fan et al., 2018; Piñero, 2015). As a result of the measurement, topographers internally compute and provide altimetric (elevation), curvature, or diopters data of the anterior or posterior sides of the cornea. Currently, most of the topographers available in clinics worldwide are Placido disk topographers as the Scheimpflug technology is newer and more expensive. Thus, it is not so well established in small or medium-sized clinics nor in countries with a more limited health care system.

Many indices or metrics using corneal topography have been proposed for the early detection of keratoconus (Alió, 2016; Prakash et al., 2012). Some of the most widely used indices are KPI (Maeda et al., 1994) and KISA (Rabinowitz and Rasheed, 1999). In recent years, work is continuing to find more reliable indices for detecting keratoconus (Alió, 2016; Ramos-López et al., 2011; Castro-Luna et al., 2020; Bühren et al., 2010), with a special focus on the early stages of this clinical condition (Ramos-López et al., 2013; Issarti et al., 2019). However, the early detection of keratoconus is still an open research problem, and new approaches have been proposed (Issarti et al., 2020; Yousefi et al., 2018; Ortiz-Toquero et al., 2020).

In ophthalmological practice, however, the reference indices are normally only those included in the topographer by default, computed internally in a black-box scheme from the raw measurements. It can be difficult for a practitioner to calculate other indices that are not directly provided by the device, either for comparison, to complement a diagnosis, or for research purposes. Therefore, this article aims to present and explain the R package rPACI (Placido Analysis of Corneal Irregularity, Ramos-López and Maldonado (2021)) and its use cases. This package was developed to facilitate the calculation and interpretation of several indices for detecting corneal irregularities, and especially keratoconus (the indices introduced in Castro-Luna et al. (2020); Ramos-López et al. (2011, 2013)). These indices proved to be effective in detecting keratoconus and early keratoconus (see those references for more details).

The main goal of **rPACI** (Ramos-López and Maldonado, 2021) is that these indices can be easily computed by an ophthalmologist, an optometrist, or any practitioner in general. To this end, several intuitive and easy-to-use functions are provided, including those that can read the data of a Placido disk topography from a file, analyze the data read and calculate the indices, return the results in a manageable format, and represent them graphically so that they are easily interpreted. To the best of the authors' knowledge, there are neither other similar R packages nor other pieces of non-commercial software sharing this goal.

As some practitioners may have difficulties in using the R syntax, we provide, in addition to the package itself, a user-friendly intuitive web application for **rPACI**, built using **shiny** (Chang et al., 2020). This graphical user interface is available to use from any web browser, and it allows the user to readily use the functionality of **rPACI** without installing R or any of its packages.

Placido corneal topography and supported file formats

The majority of topographers used in clinical practice to measure corneal topography rely on the Placido disk technology (either alone or in combination with others, such as the Scheimpflug camera) (Piñero, 2015; Samapunphong and Azar, 1998; Fan et al., 2018). In such a topographer, an illuminated pattern of concentric rings or mires (which is the actual Placido disk) is projected into the corneal surface, and its reflection on the anterior surface of the cornea is captured by a camera situated at the center of the system. The picture is then digitized along with a certain number of radii at equally spaced angles (Samapunphong and Azar, 1998; Espinosa et al., 2013).

Many manufacturers produce corneal topographers that employ the Placido disk technology. Some devices are often referred to as aberrometers or tomographers (as they also include other technologies than the Placido disk) or even keratometers or corneal analyzers. Some of these manufacturers are: CSO (Firenze, Italy), Gaush (Beijing, China), Medmont (Victoria, Australia), Nidek (Aichi, Japan), Optikon (Roma, Italy), Topcon (Tokyo, Japan), Zeiss (Oberkochen, Germany), and Ziemer (Port, Switzerland). This list has been elaborated with publicly available information or reported in scientific papers. The authors do not have any commercial or financial interest in the manufacturers reported here or in any other related companies. The **rPACI** package includes a vignette entitled "Corneal topographers and data formats", where additional information about topographers and manufacturers is provided.

There are two basic magnitudes in the Placido corneal measurement: the number of rings or mires in the actual Placido disk N_R and the number of angles (or points per ring) N_A in the digitization process. These may vary from a device to another. Typical values of N_R in commercial topographers are 20, 22, or 24, even though some topographers have as few as 10 or as many as 33. With respect to N_A, the most common values used by commercial topographers are 256 and 360, although some of them use other values.

The plain coordinates (x and y) of these points are the primary raw information measured by the Placido corneal topographers. Afterward, they usually post-process the data to obtain a third coordinate (z), yielding altimetric (elevation) or curvature data, and provide color maps and other easy-to-interpret metrics and indices to the practitioner. In some topographers, the raw measurements can be exported in a structured text file.

rPACI can read corneal topography files in two different formats by now. It will possibly be expanded in the future, allowing other formats. Both are structured plain text files. The first supported file format is basically the one employed by CSO topographers, but with some more flexibility. The second supported file format has the structure used internally by **rPACI**. These two formats are described in the next two subsections.

CSO file format

This file format is the one in which some CSO topographers (EyeTop2005, Sirius, Antares, or Osiris-T), export the raw data measurements. The corneal topography file should have the following 3-block structure:

- An optional header with metadata: patient data, optical measurements, date of exploration, etc. Its size depends on the device. For CSO topographers, the header typically has 24 lines. The **rPACI**'s reading function for these files is able to identify the header block and skip it.
- A list of size $N_R \times N_A$ with the ρ coordinate (distance to the origin in polar coordinates, measured in mm) of the digitized points, at equally spaced angles (θ, the argument in polar coordinates). The θ coordinate is inferred from the position (assuming a uniform distribution), as it is not explicitly given in the data file.
- A list of size $N_R \times N_A$ with the z coordinate, which can be altitude (elevation) or curvature, depending on the topographer and the exportation settings. These data are post-processed by the device using different algorithms, not directly measured, and the z values are not used in **rPACI**.

In the best scenario (which is not very common in the clinical practice), the N_R rings are fully available, having N_A points each, giving a total of $N_R \times N_A$ data points. However, many points are often missing, especially those corresponding to outermost rings (as they could not be digitized properly). Missing data are generally codified in the file with a specific key number (such as "-1" or "-1000"). Thus, these values do not correspond to real measured points and should be removed or substituted by `NA`s when reading the data file.

rPACI file format

This file format has been developed to directly manage the datasets handled by **rPACI**, allowing to save and read data easily. A corneal topography file with this format should have the following 2-block structure:

- An optional header of any length (its size is automatically detected and it may be missing). If the file was saved from a dataset generated by simulation, the header contains the simulation parameters.
- The data block, which consists of three separated columns (x and y coordinates of each point (in mm) and its ring index), with $N \times N_A$ lines (a line per data point), with N being the number of rings or mires and N_A being the number of points in each ring. The last column (ring index) will consist only of positive integer values, whereas the two coordinates can be any real numbers.

Since files with this format have been exported by **rPACI**, their data will consist only of full rings (either if the dataset was read from another format or generated by simulation). Thus, these files should not have any missing data.

Placido indices of corneal irregularity

Several indices of corneal irregularity can be calculated from the raw measurements of a Placido disk corneal topographer (Ramos-López et al., 2011, 2013). Many other similar indices exist (Alió, 2016; Prakash et al., 2012; Maeda et al., 1994; Rabinowitz and Rasheed, 1999; Issarti et al., 2019; Bühren et al., 2010), but they cannot be computed using just the raw data of a Placido topographer, as they require information about both surfaces of the cornea, or other parameters, besides altimetry, pachymetry, or curvature data.

The Placido indices based on the raw ring images have demonstrated their ability in diagnosing clinical diseases such as keratoconus and subclinical keratoconus, showing high accuracy (Ramos-López et al., 2011, 2013; Castro-Luna et al., 2020). For the readers' convenience, a brief summary of those indices is reported below (more details and derivations can be found in those references).

The indices available in the **rPACI** package make use only of the plain coordinates of the points (either ρ and θ in polar coordinates or x and y in Cartesian coordinates), as that is the raw information measured by a Placido corneal topographer. The post-processed data in z, if any, are discarded. Also, incomplete rings (which are normally in the periphery) are not considered for the calculation of the indices, and only the innermost, complete rings are used. Incomplete rings could be used too, but they are usually highly affected by noise and measurement errors, and their use could distort the results. Hence, after reading the data file (in any format), one has $N \times N_A$ 2D points P_j, $j = 1, \ldots, N \times N_A$, where N denotes the number of innermost complete rings ($N \leq N_R$), given in Cartesian coordinates by (x_j, y_j) or in polar coordinates by (ρ_j, θ_j).

If the data file was in CSO format, the values ρ_j are read from the file, and θ_j values are computed as $\theta_j = (2\pi/N_A)j \mod 2\pi$ (assuming these angles or arguments θ_j are equally spaced along each ring). Thus, if R_k is the k-th ring ($1 \leq k \leq N$), then

$$P_j = (\rho_j, \theta_j) \in M_k \quad \Leftrightarrow \quad j \in J_k := \{n_k, n_k + 1, ..., n_k + (N_A - 1)\}, \quad n_k = 1 + N_A(k-1).$$

The Cartesian coordinates (x_j, y_j) of point P_j can be readily computed using the variable change $x_j = \rho_j \cos(\theta_j)$, $y_j = \rho_j \sin(\theta_j)$. Figure 1 shows an example of the digitized points, both in polar and Cartesian coordinates, as given by a CSO commercial corneal topographer.

If the data file was in rPACI format, then the Cartesian coordinates (x_j, y_j) of full rings are directly read, with the same structure explained above, but without the need of applying the variable change.

The Placido irregularity indices computed by **rPACI** can be split into two categories: primary and combined indices. The primary indices measure certain geometrical properties of the data distribution. Based on them, other combined or compound indices are computed.

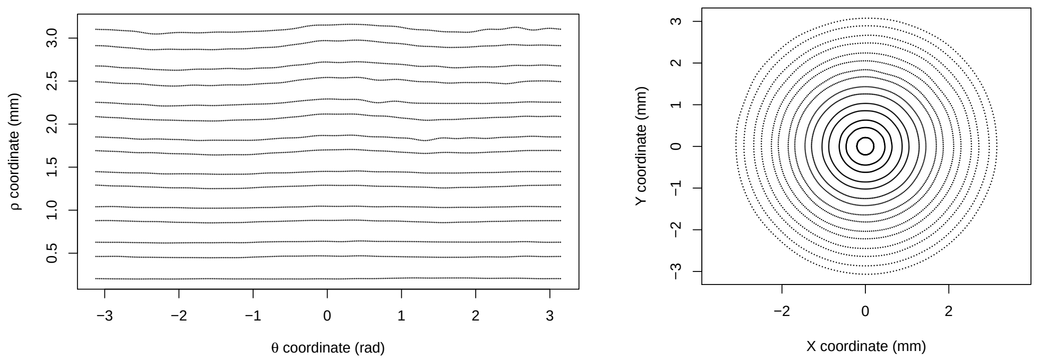


Figure 1: Corneal topography data given by a commercial Placido topographer, in polar coordinates (left) and Cartesian coordinates (right).

Primary indices

With the notation introduced before, the primary information for computing the Placido irregularity indices are the Cartesian coordinates of points belonging to full rings (no missing data):
$P_j = (x_j, y_j) \in R_k, k \in \{1, \ldots, N\}, j \in J_k$.

The first step to calculate the indices is to compute the best-fitting circle (with center C_k and (average) radius $AR(k)$) for each ring. This is done by least-squares fitting of a general circle equation to the Cartesian coordinates of data points in ring k, and then the geometrical parameters (center and radius) are computed from the coefficients (see Ramos-López et al. (2011) for more details). Similarly, one can find also best-fitting ellipses (with centers $\tilde{C}_k$ and axis ratio c_k) for each ring. Additionally, a best-fitting line can be adjusted to the coordinates of the centers C_k with slope m.

With these quantities, four indices labeled as PI_n (from "Placido Irregularity indices") and SL (from "SLope" index) can be defined:

$$PI_1 = \frac{1}{N} \max_{1 \le n,m \le N} \|C_n - C_m\|$$

corresponds to the diameter of the set of centers C_k (normalized by the total number of rings N), where $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^2$.

$$PI_2 = \frac{1}{N-1} \sum_{1 \le n \le N-1} \|C_{n+1} - C_n\|$$

corresponds to the total length of the path connecting consecutive centers.

$$PI_3 = \sqrt{\frac{1}{N} \sum_{1 \le k \le N} (c_k - \bar{c})^2}, \quad \text{where} \quad \bar{c} = \frac{1}{N} \sum_{1 \le k \le N} c_k$$

measures the variability of the axis ratios or eccentricities of the individual rings.

$$SL = |m|$$

is the absolute value of the slope of the best-fitting line to the coordinates of the centers C_k.

In order to obtain indices with values in the same range (more easily comparable and to prevent scale problems when combining them), a normalization of each primary index was performed (Ramos-López et al., 2011, 2013). After the normalization, the indices will be in general in the range $[0, 150]$, and values outside that interval are truncated. Values near zero correspond to normal corneas, whereas large values (above 70, approximately) correspond to irregular corneas. See additional details on the mathematical definition of these indices and their normalization, including the normalization coefficients, in (Ramos-López et al., 2011, 2013).

Building compound indices

The primary Placido indices described above, PI_1, PI_2, PI_3, SL, and $AR(k)$, show sensitivity to detecting various irregularities in the cornea, such as keratoconus. However, a single index does not reach sufficient accuracy in the task, and compound indices have been proposed and tested (Ramos-López et al., 2013; Castro-Luna et al., 2020) to improve the performance and precision of the

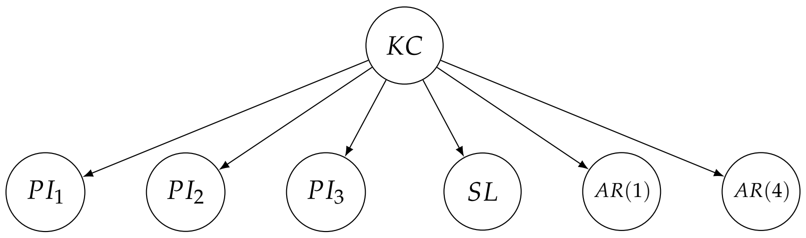


Figure 2: Structure of the Bayesian network used to compute the naïve Bayes index NBI.

individual indices. These compound indices showed a significant improvement in accuracy when predicting keratoconus. A brief description of them is given as follows (see the mentioned references for more details):

The $GLPI$ index (from Generalized Linear Placido Index) (Ramos-López et al., 2013) is a generalized linear model computed from the primary indices in the following way:

$$GLPI = 100\Phi\left(\left(-224.90 + 1.69PI_1 + 1.28PI_3 + 1.89AR(4) + 0.19SL\right)/20\right),$$

where Φ stands for the cumulative distribution function of the standard normal distribution. This corresponds to a generalized linear regression model with *probit* (Φ^{-1}) link function. Therefore, $GLPI$ has values between 0 and 100.

The compound index NBI (from Naïve Bayes Index) (Castro-Luna et al., 2020) is a Bayesian classifier. More precisely, it is a Bayesian network with the naïve Bayes structure depicted in Figure 2. This Bayesian network is a conditional linear Gaussian (CLG) network, with its root node (KC) being a discrete binary variable and the rest of the nodes being continuous. The parameters of the model were reported in Castro-Luna et al. (2020). This model can be used for predicting whether a specific cornea (whose values of the primary indices are known) is a normal cornea or a keratoconic cornea. Moreover, the probability of being one type or another can be computed as well, either using exact inference or approximate inference with algorithms such as evidence weighting, likelihood weighting, or more generally, importance sampling (Fung and Chang, 1990; Cheng and Druzdzel, 2000; Ramos-López et al., 2018). Even though the exact inference is feasible, in this case, the approximate inference is easier to implement and to generalize to more complex network structures.

In a nutshell, these algorithms consist of simulating a large number of samples from the network according to the evidence (i.e., variable values that are known a priori) and averaging their likelihoods to estimate the probability of each state of the target variable (in this case, KC).

More formally, if Y is the class variable with m states (KC in this model, with 2 states), and $\boldsymbol{X}$ is the vector of explanatory variables (PI_1, PI_2, PI_3, SL, $AR(1)$, and $AR(4)$ in this model), then the posterior probability distribution over Y can be calculated with

$$p(Y = y_j \mid \boldsymbol{X}) = \frac{p(\boldsymbol{X}, Y = y_j)}{p(\boldsymbol{X})} = \frac{p(Y = y_j)\prod_{i=1}^{n} p\left(X_i \mid Y = y_j\right)}{\sum_{j=1}^{m} p(Y = y_j)\prod_{i=1}^{n} p\left(X_i \mid Y = y_j\right)}, \quad j = 1, \ldots, m.$$

The quantities above can be estimated by drawing, according to the evidence, S independent samples $\left(y^{(s)}, \boldsymbol{x}^{(s)}\right)_{s=1}^{S} = \left(y^{(s)}, x_1^{(s)}, \ldots, x_n^{(s)}\right)_{s=1}^{S}$ from the Bayesian network and computing:

$$\hat{p}(\boldsymbol{X}, Y = y_j) = \sum_{s=1}^{S} \mathbf{1}_{y_j}\left(y^{(s)}\right) p\left(Y = y^{(s)}\right) \prod_{i=1}^{n} p\left(X_i = x_i^{(s)} \mid Y = y^{(s)}\right),$$

$$\hat{p}(\boldsymbol{X}) = \sum_{s=1}^{S} p\left(Y = y^{(s)}\right) \prod_{i=1}^{n} p\left(X_i = x_i^{(s)} \mid Y = y^{(s)}\right),$$

where $\mathbf{1}_{y_j}\left(y^{(s)}\right)$ denotes the indicator function for y_j (i.e., its values are 1 if $y_j = y^{(s)}$, and 0 otherwise). Note that in the first expression we are adding up only the terms corresponding to $Y = y_j$, whereas in the second one, terms corresponding to all samples are summed up. Thus, the probability of the class Y to take the value y_j can be estimated as:

$$\hat{p}(Y = y_j \mid \boldsymbol{X}) = \frac{\sum_{s=1}^{S} \mathbf{1}_{y_j}\left(y^{(s)}\right) p\left(Y = y^{(s)}\right) \prod_{i=1}^{n} p\left(X_i = x_i^{(s)} \mid Y = y^{(s)}\right)}{\sum_{s=1}^{S} p\left(Y = y^{(s)}\right) \prod_{i=1}^{n} p\left(X_i = x_i^{(s)} \mid Y = y^{(s)}\right)}.$$

The posterior probability estimation $\hat{p}(Y = y_j \mid \boldsymbol{X})$ can be computed efficiently with several approaches (Ramos-López et al., 2018). The R package **bnlearn** (Scutari, 2010; Scutari and Ness, 2019) includes an implementation of the likelihood weighting algorithm, which is used by **rPACI** for computing the index NBI.

The rPACI package

This section is intended to illustrate the usage of the R package **rPACI** (Ramos-López and Maldonado, 2021), which implements easy-to-use functions to read corneal topographic data and to compute the corneal irregularity indices as defined in Ramos-López et al. (2011, 2013); Castro-Luna et al. (2020) (also described in the previous section). To the best of the authors' knowledge, there are no other R packages related to the analysis of the corneal data, other indices for detecting keratoconus or other corneal diseases, or any other related topics.

The **rPACI** package includes several useful functions to analyze the corneal data of a patient and 11 example data sets, with seven of them being real measurements from a CSO topographer, and the remaining are simulated data sets. Among them, six are in CSO file format, whereas the other five are in rPACI file format. One can install and load the package as follows:

```
install.packages("rPACI", dependencies = TRUE)
library("rPACI")
```

Reading and writing data

The **rPACI** package is able to read corneal topography files in the two formats described before: CSO format and rPACI format. The package contains three functions to read data: `readCSO()`, `readrPACI()` and `readFile()`. The two former ones read data in CSO and rPACI formats, respectively, while the latter one is a wrapper function able to read both formats. In general, `readFile()` is the reading function recommended by default, as it is able to read any supported file format. These three reading functions produce a `"data.frame"` in the rPACI format, i.e., a data frame with three columns (x and y coordinates of each point and its ring index) and a row per data point. On the other hand, the package contains a function to save corneal topography data sets in the format used by **rPACI**: `writerPACI()`.

External files with a corneal topography in the format exported by some Placido disk topographers, especially those from CSO, can be loaded using the function `readCSO()`, which takes six arguments: `filepath`, the path of the file to be read; `ringsTotal`, the total number of rings available in the measurement (including incomplete rings or missing data; by default, 24); `pointsPerRings`, the number of points per ring that are digitized in the measurement (by default, 256); `ringsToUse`, the effective number of innermost rings to use (by default, 15); `onlyCompleteRings`, a logical value indicating whether to use rings with complete data only or not (by default, `TRUE`); and `NAvalues`, a numerical value or vector indicating the value(s) encoding NAs (missing data) in the file (by default, `c(-1,-1000)`).

On the other hand, external files with a corneal topography in the format exported by **rPACI** can be loaded using the `readrPACI()` function. This function has two arguments: `filepath`, the path to the file; and `sep`, the character used as column separator in the file. Finally, the general wrapper function `readFile()` internally determines the format of the specified file and applies either `readCSO()` or `readrPACI()` if possible, or else it throws an error (if none can be applied, which occurs when the file format does not fit any of these two available formats). The `readFile()` function takes the file path as a mandatory argument (`filepath`) and, optionally, any of the arguments available for the other two reading functions. In the following example, `readFile()` is used the read a sample file with the corneal topography of a normal eye (included in the package as 'N01.txt'):

```
dataset_N = readFile(system.file("extdata","N01.txt", package="rPACI"))
```

In order to save a corneal topography data set to disk, the `writerPACI()` function can be used. This function takes three arguments: `dataset`, a `data.frame` containing the corneal topography points in rPACI format; `filename`, a character string naming a file (including the extension); and `sep`, the field separator string (by default, ','). The file is saved in a structured plain text file following the rPACI format. If the data set to be saved is simulated using the `simulateData()` function (see more details on this function below), the exported file would include the parameters used for the simulation in its header. The following example exports the previously loaded object, `dataset_N`, in a text file with comma-separated values:

```
writerPACI(dataset_N, filename = "newData.txt", sep = ",")
```

Corneal analyses

The **rPACI** package includes functions to perform three different analyses: 1) analyze a single eye; 2) analyze a single eye based on repeated measures over time; and 3) analyze multiple eyes simultaneously. All analyses compute the Placido irregularity indices of the given data sets and return a table with the numerical results and a plot, which differs depending on the type of analysis performed.

To compute the Placido irregularity indices of a single eye, the `computePlacidoIndices()` can be used, which takes the object returned by `readFile()` (or, alternatively, `readCSO()` or `readrPACI()`) and two other arguments: `truncateIndicesAt150`, a logical value (by default, `TRUE`) indicating whether the primary indices should be truncated at 150 (so they are in the range 0-150) or not; and `useMaxRings`, a positive integer (by default 15) to specify the maximum number of innermost rings to use (as long as there are enough in the data set). Note that this function requires a minimum of five complete rings to work; otherwise, it will throw an error.

```
results_N = computePlacidoIndices(dataset_N)

> results_N
      Diagnose NBI GLPI PI_1 PI_2 PI_3 SL AR_1 AR_2 AR_3 AR_4 AR_5
Normal cornea   0    0   13   20   18 36   43   22   25   21   23
```

The object returned by `computePlacidoIndices()` is of class `"data.frame"` and contains 12 columns and one row (because we are analyzing only one eye). Note that the results have been rounded to integers here for a better display. The first column of the returned `"data.frame"` is the diagnosis, based on the GLPI index, which can be either '`Irregular cornea`' (GLPI$\geq$70), '`Suspect cornea`' (30 $\leq$ GLPI <70), or '`Normal cornea`' (GLPI <30). The next column is the Naïve Bayes Index (NBI), which ranges between 0 and 100 and can be interpreted as the probability of suffering from keratoconus. The remaining columns correspond to the other indices (see definitions in the previous sections).

The Placido irregularity indices can be plotted using function `plotSingleCornea()`, which takes three arguments: `dataset`, a `"data.frame"` containing the corneal topography data, i.e., the object returned by a reader function, for instance, `readFile()`; `PlacidoIndices`, a `"data.frame"` containing the computed Placido indices, i.e., the object returned by `computePlacidoIndices()`; and, optionally, `filename`, a character vector to be displayed on the plot (for instance, the filename of the corneal topography data set).

```
plotSingleCornea(dataset_N, results_N, filename = "N01.txt")
```

The result of this function can be seen in Figure 3. The left-hand side of the figure shows the input data, whereas the right-hand side shows two charts: the GLPI index plot, which visually indicates the value taken by this index on a colored scale of possible values, and the PI indices distribution, which shows the distribution of the PI_1, PI_2, PI_3 and SL indices in a boxplot placed on a scale of possible values. The color of the charts indicates whether the indices fall within the *normal cornea* region (green color), *suspicious cornea* region (orange color), or *irregular cornea* region (red color). In the example shown in Figure 3, we can clearly see that the eye is diagnosed as normal.

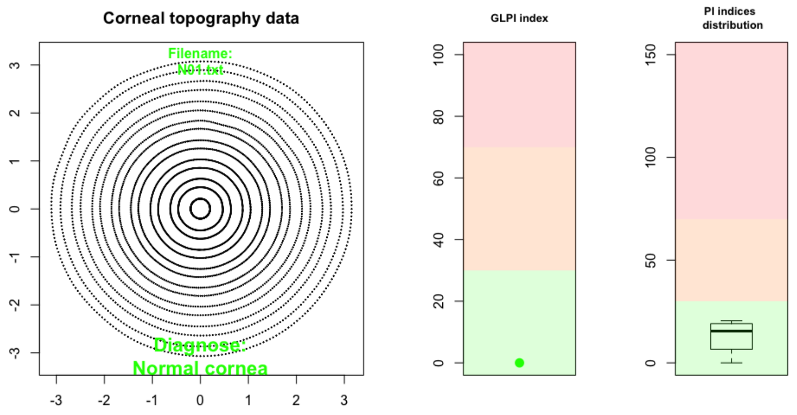


Figure 3: Results of the analysis of a normal cornea based on a single measurement.

Alternatively to using the `readFile()`, `computePlacidoIndices()` and `plotSingleCornea()` functions, the wrapper function `analyzeFile()` can be used instead, which takes 2 arguments: `path`, which is a character element indicating the location of the corneal topography file, and `drawplot`, which is a logical argument indicating whether the results should be plotted or not.

```
res_N = analyzeFile(system.file("extdata","N01.txt", package="rPACI"),
        drawplot=TRUE)
```

This function returns a "data.frame" containing the same information as the object returned by computePlacidoIndices() and, optionally, the plot returned by plotSingleCornea() if the argument drawplot is TRUE. An analogous function to analyzeFile(), called analyzeDataset(), can be used if the corneal topography data is already loaded in memory.

In order to examine the evolution of a patient's eye over time, the function analyzeEvolution() can be used. This function takes two arguments: data, which can be either 1) the path to a folder that contains corneal topography files, in any format supported by **rPACI**, or 2) a "list" containing properly formatted data (loaded from a file using the function readFile() (or readCSO() or readrPACI()), simulated using simulateData() or by other ways as long as it meets the data set requirements). If data is a path to a folder, the second argument, fileExtension, must be specified, and all the files (with the given extension) in that folder will be assumed to be corneal topography files of a patient's eye and, therefore, will be loaded.

Moreover, it is assumed that the files are arranged chronologically, i.e., the filenames should follow some date format (for instance, 'YYYY-MM-DD.txt'). On the other hand, if the data are stored in a list, it is assumed that the temporal order corresponds with the index of each data set in the list. The next example analyzes a patient's cornea at three different moments (the data sets are included in the package) :

```
analyzeEvolution(data = system.file("extdata/evolution/", package="rPACI"),
     fileExtension = 'txt')

           Diagnose NBI GLPI PI_1 PI_2 PI_3 SL AR_1 AR_2 AR_3 AR_4 AR_5 Time
1    Normal cornea 100   29  126  100    0  9    0    0    0    0    0    1
2   Suspect cornea 100   44  126  100    0 50    0    0    0    0    0    2
3 Irregular cornea 100   98  150  138    0 60    0    0    0    0    0    3
```

The analyzeEvolution() function returns a "data.frame" containing 13 columns and as many rows as files analyzed. Note that the results have been rounded to integers for a better display. The first 12 columns correspond to the diagnosis and the indices as in the object returned by computePlacidoIndices(). The last column corresponds to the time step at which the measures were taken. Moreover, two temporal plots are returned (Figure 4). The left-hand side plot shows the $GLPI$ index, represented by a red line, and the boxplots of the primary indices PI_1, PI_2, PI_3, and SL over time. The right-hand side plot shows the times series of these five indices, $GLPI$, PI_1, PI_2, PI_3, and SL, individually. Finally, both plots present a colored background, corresponding with the final diagnosis of the patient.

In this example shown in Figure 4, the results indicate that the patient's cornea was normal at time 1, suspected at time 2, and irregular at time 3.

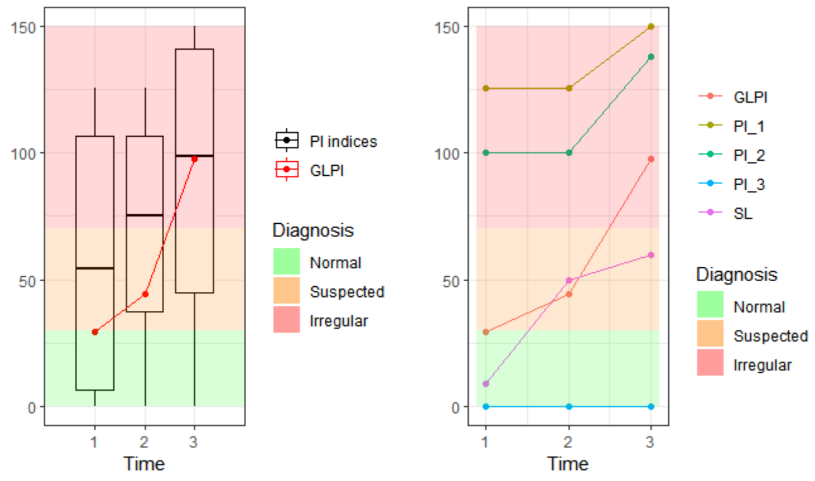


Figure 4: Results of the analysis of a cornea based on repeated measurements.

So far, analyzeFile(), analyzeDataset(), and analyzeEvolution() are intended to analyze a single cornea based on single or repeated measurements. In order to analyze multiple corneas simultaneously, the function analyzeFolder() can be used. This function takes 4 arguments: path, to indicate the location of the folder containing the files; fileExtension, which indicates the extension of the files, is set to 'txt' by default; `individualPlots`, which is an optional logical argument indicating whether the plot for each file should be displayed or not; and `summaryPlot`, which is an optional logical argument indicating whether a summary plot of all files analyzed should be displayed or not.

```
resultsAll = analyzeFolder(system.file("extdata", package="rPACI"),
individualPlots = FALSE, summaryPlot = T)

> resultsAll

          Diagnose  NBI GLPI PI_1 PI_2 PI_3  SL AR_1 AR_2 AR_3 AR_4 AR_5 Filename
3 Irregular cornea 100  100  150  150  150 142  119   72   83   73   76  K01.txt
5 Irregular cornea 100  100  128  101  119  98   57   56   47   59   51  K03.txt
4 Irregular cornea 100  100   94   85   93   4    0    0    0    0    0  K02.txt
8   Suspect cornea   0   32   43   53   68  26   39   26   31   27   30  S01.txt
7    Normal cornea   0    0    8   21   50  35   57   33   38   34   36  N02.txt
1    Normal cornea   0    0   13   20   18  36   43   22   25   21   23  ds1.txt
6    Normal cornea   0    0   13   20   18  36   43   22   25   21   23  N01.txt
2    Normal cornea   0    0    0    0    0  13    0    0    0    0    0  ds2.txt
```

This function returns a `"data.frame"` containing 13 columns and as many rows as files analyzed. Again, the results have been rounded to integers for a better display. The first 12 columns correspond to the diagnosis and the indices, as in the object returned by `computePlacidoIndices()`. The last column corresponds to the file name so that a specific patient can be easily found. To see the diagnosis for each analyzed file, the first and last columns can be selected:

```
resultsAll[,c(13,1)]

  Filename         Diagnose
3  K01.txt Irregular cornea
5  K03.txt Irregular cornea
4  K02.txt Irregular cornea
8  S01.txt   Suspect cornea
7  N02.txt    Normal cornea
1  ds1.txt    Normal cornea
6  N01.txt    Normal cornea
2  ds2.txt    Normal cornea
```

Note that the rows are sorted from '`Irregular`' to '`Normal cornea`'. If the argument `summaryPlot()` is `TRUE`, then a barplot is depicted, as shown in Figure 8 (plotted using the Shiny interface, see next section). This barplot shows the absolute frequency of each possible value of diagnosis. In this example, three eyes are diagnosed as `irregular`, one as suspect, and four as `normal`. This is an easy and straightforward way to check if any patient potentially suffers. from keratoconus.

Simulating corneal topographies

rPACI includes the function `simulateData()` that permits to simulate corneal topography data. This function has a large number of geometrical parameters that allow to generate a wide variety of corneal topographies of different clinical conditions. It produces a data set in the rPACI format, i.e., a `data.frame` with three columns (x and y coordinates of each point and its ring index) and a row per data point, according to the function parameters (by default, $15 \times 256 = 3840$ rows or data points, without missing data).

The arguments of `simulateData()` are: `rings`, the total number of rings or mires in the sample (by default, 15); `pointsPerRing`, the number of points to be sampled in each ring (by default, 256); `diameter`, the diameter of the simulated dataset in mm (typically around 8-12 mm; by default, 12 mm); `ringRadiiPerturbation`, adds a stochastical perturbation of the mires radii distribution, typically between 0 (no perturbation) and 1 (high perturbation) (by default, 0); `maximumMireDisplacement` the total mires displacement, drift or off-centering in mm, should be a reasonable number according to the diameter (by default, 0); `mireDisplacementAngle`, the direction of mires drift given as an angle in degrees, with 0 meaning positive x direction (by default, 0); `mireDisplacementPerturbation` adds a stochastical perturbation to the mires drift, typically between 0 (no perturbation) and 1 (high perturbation) (by default, 0); `ellipticAxesRatio`, the ratio or quotient between the major and minor axes of each elliptic ring, a ratio of 1 means a perfect circle (no eccentricity) (by default, 1); `ellipticRotation` direction or orientation of the elliptic mires given as an angle in degrees, with 0 meaning positive x direction (by default, 0); `overallNoise`, includes random, white noise of a certain magnitude in the Cartesian coordinates of the sampled points; the noise magnitude is relative to the diameter and the number of rings; 0 means no noise and 1 large noise (by default, 0); and finally `seed`, a seed, included for repeatability when using random perturbations.

For instance, the following piece of code simulates and depicts a corneal topography with $N = 15$ rings, $N_A = 128$ points per ring, with a diameter of 8 mm, and introducing a relatively large perturbation in the distribution of radii with `ringRadiiPerturbation = 0.7`. The results are plotted in Figure 5 (left).

```
dataset = simulateData(rings = 15, pointsPerRing = 128, diameter = 8,
                       ringRadiiPerturbation = 0.7)
plot(dataset$x,dataset$y,pch=20,cex=0.5,asp=1)
```

A highly distorted corneal topography can be generated by using the parameters related to ellipticity and mire displacement. For example, the piece of code below simulates and plots a fairly irregular corneal topography, which includes mires drift of 2mm in the direction $-30°$, with non-uniform displacements, and elliptic rings with axes ratio of 1.2 and orientated at 45°. The resulting data set is depicted in Figure 5 (right).

```
dataset = simulateData(maximumMireDisplacement = 2, mireDisplacementAngle = -30,
                       mireDisplacementPerturbation = 1.2, ellipticAxesRatio= 1.2,
                       ellipticRotation = 45)
```

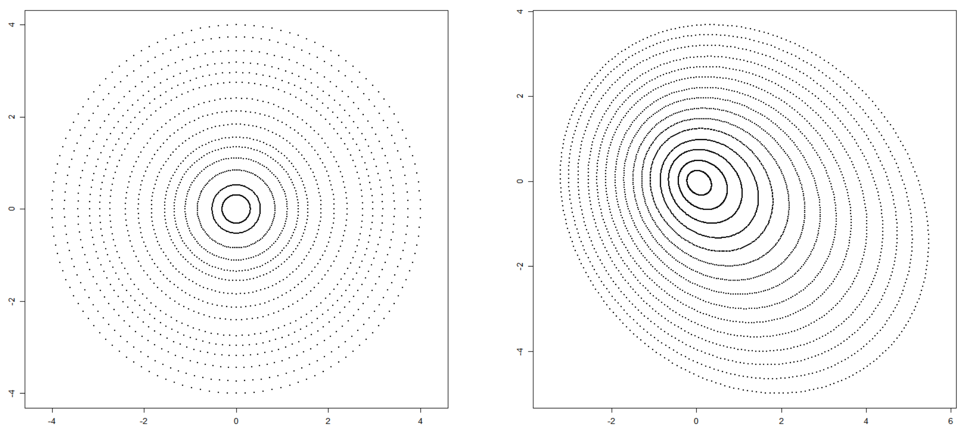


Figure 5: Simulated corneal topographies using `simulateData` with different parameters: (left) rings with non-uniform radii distribution; (right) adding mires drift and ellipticity.

More examples of using the `simulateData()` function can be found in the **rPACI** vignette "Simulating corneal datasets".

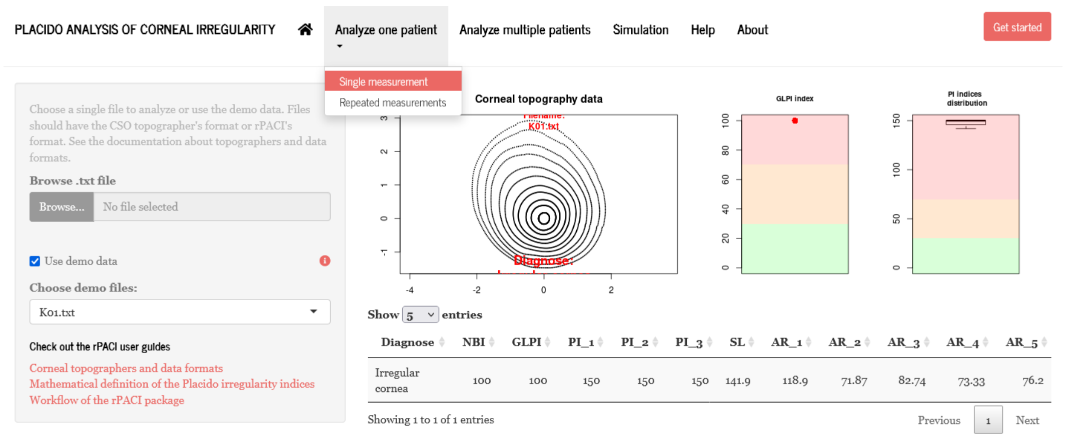
A Shiny interface to rPACI

The package presented in the previous section requires a basic knowledge of the R language. However, many practitioners might find this a serious obstacle to overcome. For this reason, a shiny interface to **rPACI** has been deployed. **Shiny** is an R package used to develop interactive web applications so that users do not need to interact with R code nor to install any software. The rPACI app can be found at `https://admaldonado.shinyapps.io/rPACI/`.

This app provides a GUI (Graphical User Interface) organized in different tabs, allowing to perform the three types of analyses above described, as well as to simulate corneal topographies. Moreover, the web app contains a 'Home' tab, to welcome users and offer a guided tour through the app; a 'Help' tab, which summarizes the app features; and an 'About' tab, which includes information about the package developers. Links to the **rPACI** vignettes, CRAN, and GitHub repositories are also available in the shiny app.

Depending on the number of corneas to analyze at a time (i.e., one or more), the user can select the '`Analyze one patient`' or '`Analyze multiple patients`' tab. If the first option is chosen, a sub-menu appears, where the user can select between '`Single measurement`' and '`Repeated measurements`'. The former choice analyzes a cornea based on a single measurement, whereas the latter analyzes a cornea based on repeated measurements over time to see its evolution. These three sections allow the users to read external files (in any format supported by **rPACI**) and also have the option to use demo data to check out how the app works.

The 'Single measurement' menu (accessed through the 'Analyze one patient' tab) contains a browser button, which allows loading one file only, and a check-box to use the available demo data sets. If the check-box is checked, a drop-down menu appears, which allows selecting a particular demo data set. This menu contains six different demo data sets, which are, in fact, real measurements. Three of them correspond to keratoconic corneas (filenames starting with 'K'), one of them to suspicious keratoconic cornea (filenames starting with 'S'), and two of them to normal corneas (filenames starting with N). When a file is selected, the app returns a plot depicted by `plotSingleCornea()` and a table containing the Placido indices. Figure 6 shows a screenshot of the app when the demo file 'K01.txt' is selected.



Choose a single file to analyze or use the demo data. Files should have the CSO topographer's format or rPACI's format. See the documentation about topographers and data formats.

Browse .txt file

Browse... No file selected

☑ Use demo data

Choose demo files:

K01.txt

Check out the rPACI user guides

Corneal topographers and data formats
Mathematical definition of the Placido irregularity indices
Workflow of the rPACI package

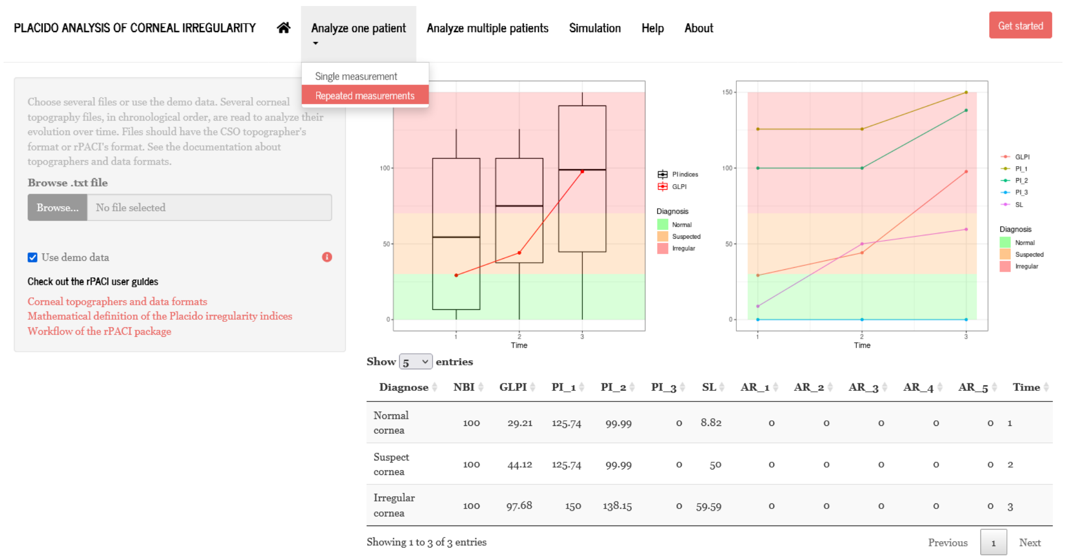
Show 5 entries

Diagnose	NBI	GLPI	PI_1	PI_2	PI_3	SL	AR_1	AR_2	AR_3	AR_4	AR_5
Irregular cornea	100	100	150	150	150	141.9	118.9	71.87	82.74	73.33	76.2

Showing 1 to 1 of 1 entries Previous 1 Next

Figure 6: Screenshot of the rPACI shiny app, performing a single measurement corneal analysis. The demo data set K01.txt is chosen.

The 'Repeated measurements' menu (accessed through the 'Analyze one patient' tab) contains a browser button that allows loading several files, whose names account for the chronological order, and a check-box to use the available demo data sets. If the check-box is ticked, the app uses a set of three simulated corneal topographies that model a patient's cornea at three different moments. This menu returns a plot depicted by `analyzeEvolution()` and a table containing the Placido indices of each data set. Figure 7 shows a screenshot of the app when the demo data is selected.



Choose several files or use the demo data. Several corneal topography files, in chronological order, are read to analyze their evolution over time. Files should have the CSO topographer's format or rPACI's format. See the documentation about topographers and data formats.

Browse .txt file

Browse... No file selected

☑ Use demo data

Check out the rPACI user guides

Corneal topographers and data formats
Mathematical definition of the Placido irregularity indices
Workflow of the rPACI package

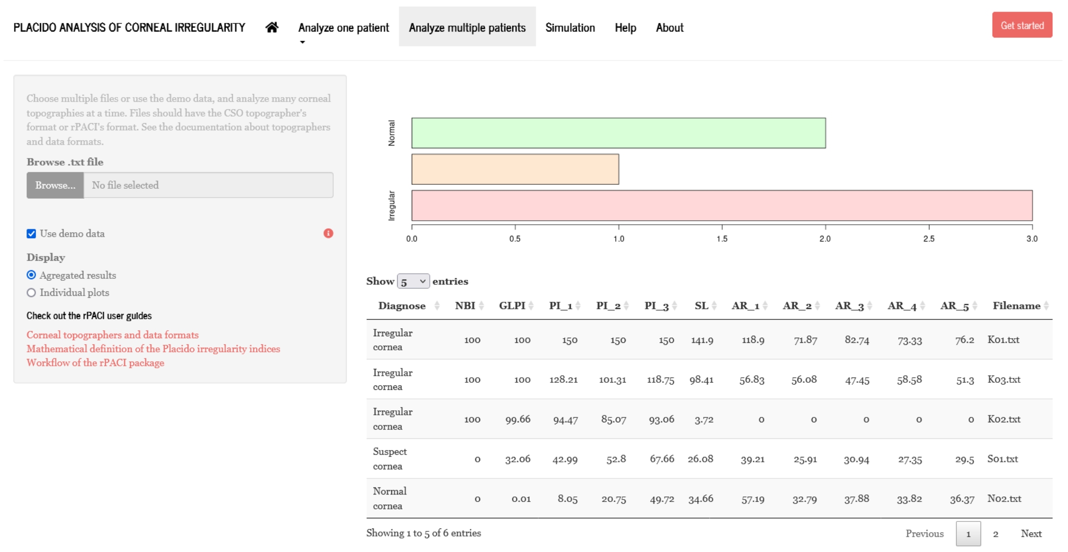
Show 5 entries

Diagnose	NBI	GLPI	PI_1	PI_2	PI_3	SL	AR_1	AR_2	AR_3	AR_4	AR_5	Time
Normal cornea	100	29.21	125.74	99.99	0	8.82	0	0	0	0	0	1
Suspect cornea	100	44.12	125.74	99.99	0	50	0	0	0	0	0	2
Irregular cornea	100	97.68	150	138.15	0	59.59	0	0	0	0	0	3

Showing 1 to 3 of 3 entries Previous 1 Next

Figure 7: Screenshot of the rPACI shiny app, performing a repeated measurements corneal analysis. The available demo data is chosen for illustrative purposes.

The 'Analyze multiple patients' tab contains a browser button that allows loading several files, a check-box to use the available demo data sets, and two mutually exclusive radio buttons to chose the display of the results. If the check-box is marked, the analysis is performed on six available demo data sets. The radio buttons allow visualizing either the aggregated results (displaying a barplot of the diagnosis values) or the individual plots of each file, in which case a drop-down menu that allows selecting a specific case appears. Moreover, this menu returns a table containing the Placido indices of the selected files. Note that the table is interactive, as the rows can be sorted according to a specific column, and the number of entries to display can be changed. Figure 8 shows a screenshot of the app when the demo data is selected.



Choose multiple files or use the demo data, and analyze many corneal topographies at a time. Files should have the CSO topographer's format or rPACI's format. See the documentation about topographers and data formats.

Browse .txt file: No file selected

☑ Use demo data

Display
[✓ Agregated results]
Individual plots

Check out the rPACI user guides
Corneal topographers and data formats
Mathematical definition of the Placido irregularity indices
Workflow of the rPACI package

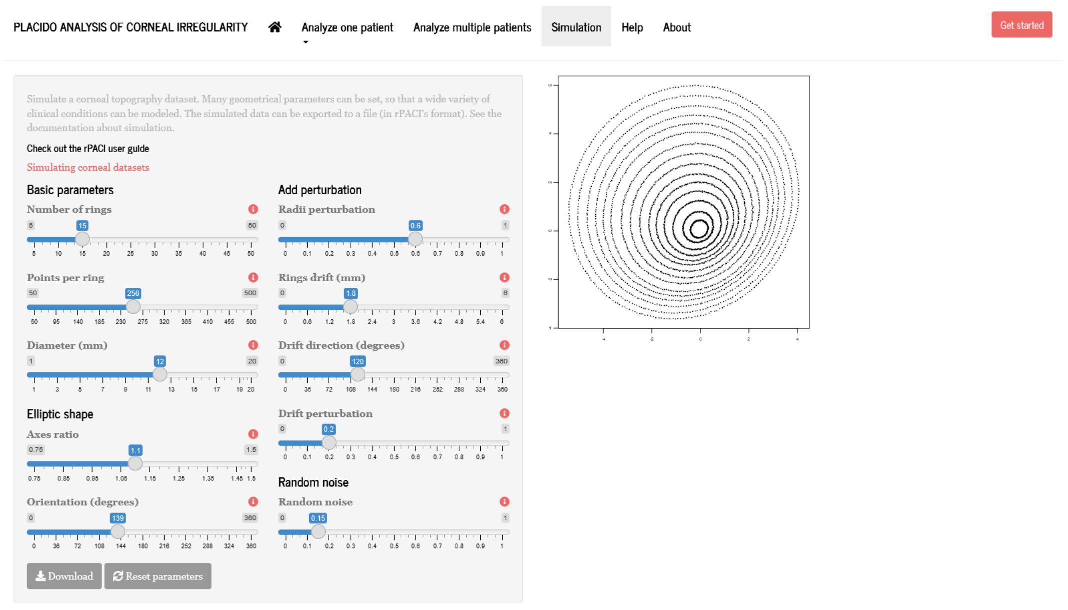
Show 5 entries

Diagnose	NBI	GLPI	PI_1	PI_2	PI_3	SL	AR_1	AR_2	AR_3	AR_4	AR_5	Filename
Irregular cornea	100	100	150	150	150	141.9	118.9	71.87	82.74	73.33	76.2	K01.txt
Irregular cornea	100	100	128.21	101.31	118.75	98.41	56.83	56.08	47.45	58.58	51.3	K03.txt
Irregular cornea	100	99.66	94.47	85.07	93.06	3.72	0	0	0	0	0	K02.txt
Suspect cornea	0	32.06	42.99	52.8	67.66	26.08	39.21	25.91	30.94	27.35	29.5	S01.txt
Normal cornea	0	0.01	8.05	20.75	49.72	34.66	57.19	32.79	37.88	33.82	36.37	N02.txt

Showing 1 to 5 of 6 entries

Figure 8: Screenshot of the rPACI shiny app, performing an analysis on multiple corneas simultaneously. The available demo data is chosen for illustrative purposes.

Finally, the 'Simulation' tab allows simulating a corneal topography data set, specifying up to ten different parameters. The simulated data set can be downloaded using the button designed for that purpose, and the parameters can be reset to the default values. If the simulated data set is downloaded, the file is saved as a '.txt' file in rPACI format, containing a header with the chosen simulation parameters. This file can later be loaded into the app to perform any kind of analysis. Figure 9 shows a screenshot of the app performing a corneal topography simulation.



Simulate a corneal topography dataset. Many geometrical parameters can be set, so that a wide variety of clinical conditions can be modeled. The simulated data can be exported to a file (in rPACI's format). See the documentation about simulation.

Check out the rPACI user guide
Simulating corneal datasets

Basic parameters
Number of rings: 15 (5–50)
Points per ring: 256 (50–500)
Diameter (mm): 12 (1–20)

Elliptic shape
Axes ratio: 1.1 (0.75–1.5)
Orientation (degrees): 139 (0–360)

Add perturbation
Radii perturbation: 0.6 (0–1)
Rings drift (mm): 1.8 (0–6)
Drift direction (degrees): 120 (0–360)
Drift perturbation: 0.2 (0–1)

Random noise
Random noise: 0.15 (0–1)

Figure 9: Screenshot of the 'Simulation' tab in the rPACI shiny app.

Summary

Diagnose of corneal diseases, especially keratoconus, is still an important research and clinical problem. Most of the corneal topographers in clinics are Placido disk devices, and several keratoconus indices for these devices exist. In this paper, we have presented the R package **rPACI**, whose aim is to provide practitioners or researchers in the field of ophthalmology with several easy-to-use functions to handle corneal data as provided by a Placido disk corneal topographer. This package includes several indices for detecting keratoconus or other irregularities available in the literature, which proved to be useful and accurate in the diagnosis.

As the main goal is to facilitate the users the computation and use of the indices, the R package **rPACI** was designed to be as easy to use as possible and provide results (data tables and graphics) that are effortless to interpret. Additionally, a **shiny** web app was developed and openly deployed (`https://admaldonado.shinyapps.io/rPACI/`) so that **rPACI** can be used in any web browser, in a truly user-friendly graphical interface, without the need of installing R or **rPACI** or writing any R code.

In the future, we hope other indices may be included in this package or in other new R packages, facilitating even further their use, improvement, or validation by practitioners or researchers in the topic and contributing to new advances in the field. Also, we would like this package and its **shiny** GUI to serve as an inspiration for other researchers in the field of healthcare to make their research results more accessible to clinicians and other researchers.

Acknowledgments

D. R-L acknowledges the support from the research group FQM-229 (Junta de Andalucía), from the 'Campus de Excelencia Internacional del Mar' CEIMAR (University of Almería) and from the Center for Development and Transfer of Mathematical Research to Industry CDTIME (University of Almería). This research is part of Project PID2019-106758GB-C32 funded by MCIN/AEI/10.13039/501100011033, FEDER "Una manera de hacer Europa" funds. This research is also partially funded by Junta de Andalucía Project PY20_00091. A.D.M. thanks the support by Junta de Andalucía through Grant DOC_00358.

Bibliography

P. A. Accardo and S. Pensiero. Neural network-based system for early keratoconus detection from corneal topography. *Journal of Biomedical Informatics*, 35:151–159, 2002. URL `https://doi.org/10.1016/S1532-0464(02)00513-0`. [p321]

J. L. Alió. *Keratoconus: recent advances in diagnosis and treatment*. Springer, 2016. URL `https://doi.org/10.1007/978-3-319-43881-8`. [p321, 323]

J. L. Alió and M. H. Shabayek. Corneal higher order aberrations: a method to grade keratoconus. *Journal of Refractive Surgery*, 22:539–545, 2006. URL `https://doi.org/10.3928/1081-597X-20060601-05`. [p321]

W. Alkhaldi, D. R. Iskander, A. M. Zoubir, and M. J. Collins. Enhancing the standard operating range of a Placido disk videokeratoscope for corneal surface estimation. *IEEE Transactions on Biomedical Engineering*, 56(3):800–809, 2009. URL `https://doi.org/10.1109/TBME.2008.2005997`. [p321]

J. Bühren, D. Kook, G. Yoon, and T. Kohnen. Detection of subclinical keratoconus by using corneal anterior and posterior surface aberrations and thickness spatial profiles. *Investigative Ophthalmology and Visual Science*, 51:3424–3432, 2010. URL `https://doi.org/10.1167/iovs.09-4960`. [p321, 323]

G. M. Castro-Luna, A. Martínez-Finkelshtein, and D. Ramos-López. Robust keratoconus detection with Bayesian network classifier for Placido-based corneal indices. *Contact Lens and Anterior Eye*, 43 (4):366–372, 2020. URL `https://doi.org/10.1016/j.clae.2019.12.006`. [p321, 323, 324, 325, 326]

W. Chang, J. Cheng, J. J. Allaire, Y. Xie, and J. McPherson. *Shiny: web application framework for R*, 2020. URL `https://CRAN.R-project.org/package=shiny`. R package version 1.5.0. [p322]

J. Cheng and M. J. Druzdzel. AIS-BN: an adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000. URL `https://doi.org/10.1613/jair.764`. [p325]

U. de-Sanctis, C. Loiacono, L. Richiardi, D. Turco, B. Mutani, and F. M. Grignolo. Sensitivity and specificity of posterior corneal elevation measured by Pentacam in discriminating keratoconus/subclinical keratoconus. *Ophthalmology*, 115:1534–1539, 2008. URL https://doi.org/10.1016/j.ophtha.2008.02.020. [p321]

J. Espinosa, D. Mas, J. Pérez, and A. B. Roig. Open-access operating algorithms for commercial videokeratographer and improvement of corneal sampling. *Applied Optics*, 52(7):C24–C29, 2013. URL https://doi.org/10.1364/AO.52.000C24. [p322]

R. Fan, T. C. Chan, G. Prakash, and V. Jhanji. Applications of corneal topography and tomography: A review. *Clinical & Experimental Ophthalmology*, 46(2):133–146, 2018. URL https://doi.org/10.1111/ceo.13136. [p321, 322]

R. Fung and K. C. Chang. Weighting and integrating evidence for stochastic simulation in Bayesian networks. *Machine Intelligence and Pattern Recognition*, 10:209–219, 1990. URL https://doi.org/10.1016/B978-0-444-88738-2.50023-3. [p325]

I. Issarti, A. Consejo, M. Jiménez-García, S. Hershko, C. Koppen, and J. J. Rozema. Computer aided diagnosis for suspect keratoconus detection. *Computers in biology and medicine*, 109:33–42, 2019. URL https://doi.org/10.1016/j.compbiomed.2019.04.024. [p321, 323]

I. Issarti, A. Consejo, M. Jiménez-García, E. O. Kreps, C. Koppen, and J. J. Rozema. Logistic index for keratoconus detection and severity scoring (Logik). *Computers in Biology and Medicine*, 122:103809, 2020. URL https://doi.org/10.1016/j.compbiomed.2020.103809. [p321]

L. Llorente, S. Barbero, J. Merayo, and S. Marcos. Total and corneal optical aberrations induced by laser in situ keratomileusis for hyperopia. *Journal of Refractive Surgery*, 20:203–216, 2004. URL https://doi.org/10.3928/1081-597X-20040501-03. [p321]

N. Maeda, S. D. Klyce, M. K. Smolek, and H. W. Thompson. Automated keratoconus screening with corneal topography analysis. *Investigative Ophthalmology and Visual Science*, 35:2749–2757, 1994. [p321, 323]

L. J. Maguire and W. M. Bourne. Corneal topography of early keratoconus. *American Journal of Ophthalmology*, 108:107–112, 1989. URL https://doi.org/10.1016/0002-9394(89)90001-9. [p321]

A. Martínez-Abad and D. P. Piñero. New perspectives on the detection and progression of keratoconus. *Journal of Cataract & Refractive Surgery*, 43(9):1213–1227, 2017. URL https://doi.org/10.1016/j.jcrs.2017.07.021. [p321]

S. Ortiz-Toquero, I. Fernández, and R. Martín. Classification of keratoconus based on anterior corneal high-order aberrations: a cross-validation study. *Optometry and Vision Science*, 97:169–177, 2020. URL https://doi.org/10.1097/OPX.0000000000001489. [p321]

D. P. Piñero. Technologies for anatomical and geometric characterization of the corneal structure and anterior segment: A review. *Seminars in Ophthalmology*, 30(3):161–170, 2015. URL https://doi.org/10.3109/08820538.2013.835844. [p321, 322]

G. Prakash, A. Agarwal, A. I. Mazhari, G. Kumar, P. Desai, D. A. Kumar, and S. Jacob. A new, pachymetry-based approach for diagnostic cutoffs for normal, suspect and keratoconic cornea. *Eye (London)*, 26:650–657, 2012. URL https://doi.org/10.1038/eye.2011.365. [p321, 323]

Y. S. Rabinowitz. Keratoconus. *Survey of Ophthalmology*, 42:297–319, 1998. URL https://doi.org/10.1016/s0039-6257(97)00119-7. [p321]

Y. S. Rabinowitz and K. Rasheed. KISA % index: a quantitative videokeratography algorithm embodying minimal topographic criteria for diagnosing keratoconus. *Journal of Cataract and Refractive Surgery*, 25:1327–1335, 1999. URL https://doi.org/10.1016/s0886-3350(99)00195-9. [p321, 323]

D. Ramos-López and A. D. Maldonado. *rPACI: Placido Analysis of Corneal Irregularity*, 2021. URL https://CRAN.R-project.org/package=rPACI. R package version 0.2.1. [p321, 326]

D. Ramos-López, A. Martínez-Finkelshtein, G. M. Castro-Luna, D. P. Piñero, and J. L. Alió. Placido-based indices of corneal irregularity. *Optometry and Vision Science*, 88(10):1220–1231, 2011. URL https://doi.org/10.1097/OPX.0b013e3182279ff8. [p321, 323, 324, 326]

D. Ramos-López, A. Martínez-Finkelshtein, G. M. Castro-Luna, N. Burguera-Gimenez, A. Vega-Estrada, D. P. Piñero, and J. L. Alió. Screening subclinical keratoconus with Placido-based corneal indices. *Optometry and Vision Science*, 90(4):335–343, 2013. URL https://doi.org/10.1097/OPX.0b013e3182279ff8. [p321, 323, 324, 325, 326]

D. Ramos-López, A. R. Masegosa, A. Salmerón, R. Rumí, H. Langseth, T. D. Nielsen, and A. L. Madsen. Scalable importance sampling estimation of Gaussian mixture posteriors in Bayesian networks. *International Journal of Approximate Reasoning*, 100:115–134, 2018. URL https://doi.org/10.1016/j.ijar.2018.06.004. [p325, 326]

R. H. Rand, H. C. Howland, and R. A. Applegate. Mathematical model of a Placido disk keratometer and its implications for recovery of corneal topography. *Optometry and Vision Science*, 74:926–930, 1997. URL https://doi.org/10.1097/00006324-199711000-00026. [p321]

J. B. Randleman, B. Russel, M. A. Ward, K. P. Thompson, and R. D. Stulting. Risk factors and prognosis for corneal ectasia after LASIK. *Ophthalmology*, 110:267–175, 2003. URL https://doi.org/10.1016/s0161-6420(02)01727-x. [p321]

S. A. Read, M. J. Collins, D. R. Iskander, and B. A. Davis. Corneal topography with Scheimpflug imaging and videokeratography: comparative study of normal eyes. *Journal of Cataract and Refractive Surgery*, 35(6):1072–1081, 2009. URL https://doi.org/10.1016/j.jcrs.2009.01.020. [p321]

J. J. Rowsey, A. E. Reynolds, and R. Brown. Corneal topography: corneascope. *Archives of Ophthalmology*, 99(6):1093–1100, 1981. URL https://doi.org/10.1001/archopht.1981.03930011093022. [p321]

S. Samapunphong and D. Azar. Placido and elevation-based corneal topography. A review. *Ophthalmology Clinics of North America*, 11(3):311–329, 1998. URL https://doi.org/10.1016/S0896-1549(05)70059-6. [p321, 322]

M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35 (3):1–22, 2010. URL https://doi.org/10.18637/jss.v035.i03. [p326]

M. Scutari and R. Ness. *bnlearn: Bayesian network structure learning, parameter learning and inference*, 2019. URL https://CRAN.R-project.org/package=bnlearn. R package version 4.5. [p326]

S. Yousefi, E. Yousefi, H. Takahashi, T. Hayashi, H. Tampo, S. Inoda, Y. Arai, and P. Asbell. Keratoconus severity identification using unsupervised machine learning. *PLOS ONE*, 13(11):1–11, 2018. URL https://doi.org/10.1371/journal.pone.0205998. [p321]

Darío Ramos-López
Department of Mathematics
University of Almería
Carretera Sacramento s/n, 04120 La Cañada, Almería
Spain
ORCiD: https://orcid.org/0000-0002-6127-6559
dramoslopez@ual.es

Ana D. Maldonado
Department of Mathematics
University of Almería
Carretera Sacramento s/n, 04120 La Cañada, Almería
Spain
ORCiD: https://orcid.org/0000-0001-8253-2526
ana.d.maldonado@ual.es

MAINT.Data: Modelling and Analysing Interval Data in R

by A. Pedro Duarte Silva, Paula Brito, Peter Filzmoser and José G. Dias

Abstract We present the CRAN R package **MAINT.Data** for the modelling and analysis of multivariate interval data, i.e., where units are described by variables whose values are intervals of $\mathbb{R}$, representing intrinsic variability. Parametric inference methodologies based on probabilistic models for interval variables have been developed, where each interval is represented by its midpoint and log-range, for which multivariate Normal and Skew-Normal distributions are assumed. The intrinsic nature of the interval variables leads to special structures of the variance-covariance matrix, which are represented by four different possible configurations. **MAINT.Data** implements the proposed methodologies in the S4 object system, introducing a specific data class for representing interval data. It includes functions and methods for modelling and analysing interval data, in particular maximum likelihood estimation, statistical tests for the different configurations, (M)ANOVA and Discriminant Analysis. For the Gaussian model, Model-based Clustering, robust estimation, outlier detection and Robust Discriminant Analysis are also available.

Introduction

In classical statistics and multivariate data analysis, the basic units under analysis are single individuals, described by numerical and/or categorical variables, each individual taking one single value for each variable. For instance, a specific football player may be described by his age, height, weight, goals marked, nationality; a specific passenger by his/her gender, age, destination, weight of luggage, etc. Data are organised in a data-array, where each cell (i, j) contains the value of variable j for individual i.

It is however often the case that the data under analysis are not single observations, but rather sets of values, either related to groups of units gathered on the basis of some common properties, or observed repeatedly over time or under different specific conditions. The classical framework is then somehow restricted to take into account variability inherent to such data. This is the case when we are interested in describing football teams and not each specific player, or flights and not each particular passenger. The same issue often arises in official statistics analysis. Whether it is for the analysis' purposes, or for confidentiality reasons, individual data – here usually called "microdata" – is gathered into more general data arrays, related to parishes, counties, socio-economical groups, etc. – the so-called "macro-data". Internal variability should also be considered when the focus of the analysis lies in concepts (i.e., all elements sharing a given set of defining properties) rather than in a single specimen – whether it is a plant species (and not the specific plant I hold in my hand), a model of car (and not the particular one I am driving), etc. Another pertinent case arises when we are facing huge amounts of data, recorded in very large databases, and elements of interest are not the individual records but some second-level entities. For instance, in a database of a hypermarket purchases, we are surely more interested in describing the behaviour of some client (or some pre-defined class or group of clients) rather than each purchase by itself. The analysis requires then that the purchase data for each person (or group) be somehow aggregated to obtain the information of interest; here again the observed variability for each client or within each group is of utmost importance, and cannot be retained by summary statistics.

Symbolic Data Analysis (see e.g. Diday and Noirhomme-Fraiture (2008), Brito (2014)) provides a framework where the variability observed may effectively be considered in the data representation, and methods are developed that take that into account. To describe groups of individuals or concepts, new variable types may now assume other forms of realisations, which allow taking intrinsic variability into account. They may take the form of finite sets, intervals or distributions. In recent years, different approaches have been investigated and many methods proposed for the analysis of such symbolic data, and for the design of a symbolic counterpart of statistical multivariate data analysis methods. Most existing methods for the analysis of such data rely however on non-parametric descriptive approaches. Among these, interval data is by far the most investigated data type and for which more methods have been developed.

In Brito and Duarte Silva (2012), parametric inference methodologies based on probabilistic models for interval variables are developed where each interval is represented by its midpoint and log-range, for which multivariate Normal and Skew-Normal (Azzalini and Dalla Valle, 1996) distributions are assumed. The intrinsic nature of the interval variables leads to special structures of the variance-covariance matrix, which are represented by different possible configurations.

It should be noticed that we are modelling interval-valued variables, i.e. variables whose observed values are intervals, and not single-valued real variables. For this reason, they should not be confused with real-valued variables whose values are restricted to some intervals. Data structures for this latter type are available in some R packages such as **survreg** (Hubeaux and Rufibach, 2015) or **crch** (Messner et al., 2019), but they obviously do not apply in our context.

In this paper, we present the package **MAINT.Data** (Duarte Silva and Brito, 2021), which implements the proposed methodologies in R (R Core Team, 2021). **MAINT.Data** is built using S4 classes and methods, introducing a specific data class for representing interval data. Functions for aggregating microdata into interval data objects are also provided. **MAINT.Data** includes functions and methods for modelling and analysing interval data, in particular maximum likelihood estimation and statistical tests for the different considered configurations. Methods for (M)ANOVA (Brito and Duarte Silva, 2012) and Discriminant Analysis (Duarte Silva and Brito, 2015) of this data class are also provided. For the Gaussian model, Model-based Clustering (Brito et al., 2015), robust estimation and outlier detection (Duarte Silva et al., 2018) are implemented; corresponding methods for Robust Discriminant Analysis are also available.

Multivariate analysis of interval-valued data has been addressed from different perspectives, as Clustering (see, e.g., De Carvalho et al. (2006); De Carvalho and Lechevallier (2009)), Principal Component Analysis (PCA) (see, e.g. Douzal-Chouakria et al. (2011); Le-Rademacher and Billard (2012)), Discriminant Analysis (Duarte Silva and Brito, 2015; Ramos-Guajardo and Grzegorzewski, 2016), Regression Analysis (Dias and Brito, 2017; Lima Neto and De Carvalho, 2008, 2010, 2011), etc. For a survey the reader may refer to Brito (2014). Those are mostly non-parametric exploratory methodologies; recent approaches based on parametric models have also been proposed in Brito and Duarte Silva (2012), Le-Rademacher and Billard (2011), and Lima Neto and De Carvalho (2011).

Many of the methods mentioned above for analysing interval-valued data may be found in R packages, namely **symbolicDA** (Dudek et al., 2019), (general multivariate data analysis/machine learning approaches, e.g. PCA, Discriminant Analysis, Multidimensional Scaling, Clustering), **RSDA** (Rodriguez, 2021) (mainly classification and linear models), **iRegression** (Lima Neto et al., 2016) (Regression) and **GPCSIV** (Brahim and Makosso-Kallyth, 2013) (PCA). We note that most of these packages implement non-parametric methods, an exception being **iRegression** which comprehends regression based on the parametric approach proposed in Lima Neto and De Carvalho (2011). To the best of our knowledge, no other implementations of parametric approaches for the (multivariate) analysis of interval-valued data are publicly available.

The remainder of the paper is organised as follows. In the next section, we introduce interval data array and fix notation. Section **Models and estimation** presents the proposed models and the estimation of corresponding parameters. Section **Multivariate analysis** develops multivariate analysis methods based on those models. Section **Package** discusses the main structure and technical implementation of the **MAINT.Data** package. In Section **Applications**, two applications illustrate the use of the package and its functionalities. Finally, Section **Summary** concludes the paper, pointing out perspectives for future developments.

Interval data

Let $S = \{s_1, \ldots, s_n\}$ be the set of n units under analysis. An interval variable is defined by an application

$$Y : S \to T \text{ such that } s_i \to Y(s_i) = [l_i, u_i]$$

where T is the set of intervals of an underlying set $O \subseteq \mathbb{R}$. Let I be an $n \times p$ matrix containing the values of p interval variables on S. Each $s_i \in S$ is then represented by a p-dimensional vector of intervals, $I_i = (I_{i1}, \ldots, I_{ip}), i = 1, \ldots, n$, with $I_{ij} = [l_{ij}, u_{ij}]$, with $u_{ij} \geq l_{ij}, j = 1, \ldots, p$ (see Table 1). The models considered in **MAINT.Data** assume all intervals are non-degenerate, i.e., $u_{ij} > l_{ij}, j = 1, \ldots, p, i = 1, \ldots, n$.

The value of an interval variable Y_j for each $s_i \in S$ is defined by the lower and upper bounds l_{ij} and u_{ij} of $I_{ij} = Y_j(s_i)$, here assumed to be strictly different (i.e. degenerate intervals are not considered in this framework). For modelling purposes, however, an alternative parametrisation that consists in representing $Y_j(s_i)$ by the MidPoint $c_{ij} = \dfrac{l_{ij} + u_{ij}}{2}$ and Log-Range $r^*_{ij} = \ln(u_{ij} - l_{ij})$ of I_{ij} is often adopted.

We note that the interval-valued data considered here do not represent uncertainty, but rather intrinsic variability. Such interval data may occur directly, or result from the aggregation of microdata. "Native" interval data are common e.g. in Botany and Zoology, one example being the length of the stem of a given plant species, which of course varies from specimen to specimen. The aggregation of

	Y_1	$\dots$	Y_j	$\dots$	Y_p
s_1	$[l_{11}, u_{11}]$	$\dots$	$[l_{1j}, u_{1j}]$	$\dots$	$[l_{1p}, u_{1p}]$
$\dots$	$\dots$		$\dots$		$\dots$
s_i	$[l_{i1}, u_{i1}]$	$\dots$	$[l_{ij}, u_{ij}]$	$\dots$	$[l_{ip}, u_{ip}]$
$\dots$	$\dots$		$\dots$		$\dots$
s_n	$[l_{n1}, u_{n1}]$	$\dots$	$[l_{nj}, u_{nj}]$	$\dots$	$[l_{np}, u_{np}]$

Table 1: Matrix I of interval data

microdata from potentially large databases also provides interval data, when individual numerical records are combined at the required level of granularity leading to a range of values representing the underlying variability. An example of such a case is the aggregation of the values of single purchases say, in the Bakery and Dairy section of a supermarket, for each client, during a year – we then obtain, for each client and for each supermarket section, an interval representing the variability of purchase values. Such aggregations are usually based on observed minima and maxima, but specific quantiles may also be considered for this purpose.

Models and estimation

Models specification

In Brito and Duarte Silva (2012), parametric models for interval data, relying on multivariate Normal or Skew-Normal distributions for the MidPoints and Log-Ranges of the interval-valued variables have been proposed.

The Gaussian model consists in assuming a joint multivariate Normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for the MidPoints C and the logs of the Ranges R^*, with $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_C^t\ \boldsymbol{\mu}_{R^*}^t\right]^t$ and $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{CC} & \boldsymbol{\Sigma}_{CR^*} \\ \boldsymbol{\Sigma}_{R^*C} & \boldsymbol{\Sigma}_{R^*R^*} \end{pmatrix}$ where $\boldsymbol{\mu}_C$ and $\boldsymbol{\mu}_{R^*}$ are p-dimensional column vectors of the mean values of, respectively, the MidPoints and Log-Ranges, and $\boldsymbol{\Sigma}_{CC}, \boldsymbol{\Sigma}_{CR^*}, \boldsymbol{\Sigma}_{R^*C}$ and $\boldsymbol{\Sigma}_{R^*R^*}$ are $p \times p$ matrices with their variances and covariances. This model has the advantage of allowing for a straightforward application of classical multivariate methods.

Given that the MidPoint C_{ij} and the Log-Range R^*_{ij} of the value of an interval variable $Y_j(s_i)$ are related to the same variable, they should, therefore, be considered together and their relation taken into account by appropriate configurations of the global covariance matrix. Intermediate parametrisations between the non-restricted and the non-correlation setup considered for real-valued data are, therefore, relevant for the specific case of interval data.

The most general formulation allows for non-zero correlations among all MidPoints and Log-Ranges (configuration 1); in another setup, interval variables Y_j are independent, but for each variable, the MidPoint may be correlated with its Log-Range (configuration 2); a third situation allows for MidPoints (respectively, Log-Ranges) of different variables to be correlated, but no correlation between MidPoints and Log-Ranges is allowed (configuration 3); finally, all MidPoints and Log-Ranges may be uncorrelated, both among themselves and between each other (configuration 4). Table 2 summarizes the different considered configurations. We note that from the normality assumption it follows that, in this particular framework, imposing non-correlations with Log-Ranges is equivalent to imposing non-correlations with Ranges.

Configuration	Characterization	$\boldsymbol{\Sigma}$
C1	Not restricted	Not restricted
C2	Y_j's not correlated	$\boldsymbol{\Sigma}_{CC}, \boldsymbol{\Sigma}_{CR^*} = \boldsymbol{\Sigma}_{R^*C}$, $\boldsymbol{\Sigma}_{R^*R^*}$ all diagonal
C3	C's not-correlated with R^*'s	$\boldsymbol{\Sigma}_{CR^*} = \boldsymbol{\Sigma}_{R^*C} = \mathbf{0}$
C4	All C's and R^*'s are not-correlated	$\boldsymbol{\Sigma}$ diagonal

Table 2: Different cases for the variance-covariance matrix.

It should be remarked that for configurations C2, C3 and C4, $\boldsymbol{\Sigma}$ can be written as a block diagonal matrix, after a possible rearrangement of rows and columns.

In Brito and Duarte Silva (2012) another configuration has been considered, where MidPoints (respectively, Log-Ranges) of different variables may be correlated, the MidPoint of each variable may be correlated with its Log-Range, but no correlation between Midpoints and Log-Ranges of different variables is allowed. However, this case seems less natural, and leads to computational difficulties, since $\mathbf{\Sigma}$ can no longer be written as a block diagonal matrix, and, therefore, it has not been used in subsequent studies.

The Gaussian model has many advantages, which explains its generalized use in multivariate data analysis; in particular, it allows for a direct modelling of the covariance structure between the variables. Nevertheless, it does present some limitations, namely the fact that it imposes a symmetrical distribution on the MidPoints and a specific relation between mean, variance and skewness for the Ranges. A more general model that overcomes these limitations may be obtained by considering the family of Skew-Normal distributions (see, for instance, Azzalini (1985); Azzalini and Dalla Valle (1996)). The Skew-Normal generalizes the Gaussian distribution by introducing an additional shape parameter, while trying to preserve some of its mathematical properties.

The density of a q-dimensional Skew-Normal distribution is given by

$$f(x;\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\Omega}) = 2\phi_q(\mathbf{x}-\boldsymbol{\xi};\boldsymbol{\Omega})\Phi(\boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}(\mathbf{x}-\boldsymbol{\xi})), \mathbf{x} \in I\!R^q \tag{1}$$

where now $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$ are q-dimensional vectors, $\boldsymbol{\Omega}$ is a symmetric $q \times q$ positive-definite matrix, $\boldsymbol{\omega}$ is a diagonal matrix formed by the square-roots of the diagonal elements of $\boldsymbol{\Omega}$, ϕ_q is the density of a $N_q(0,\boldsymbol{\Omega})$ and Φ is the distribution function of a standard Gaussian variable.

Notice that the Skew-Normal model encompasses mixed models with marginal Normal random variables, for which the corresponding shape parameter is null.

The mean vector, variance-covariance matrix, and skewness coefficients of a q-dimensional Skew-Normal distribution are given by (see Azzalini (2005))

$$\boldsymbol{\mu} = E(\boldsymbol{X}) = \boldsymbol{\xi} + \boldsymbol{\omega}\boldsymbol{\mu}_Z \tag{2}$$

$$\boldsymbol{\Sigma} = Var(\boldsymbol{X}) = \boldsymbol{\Omega} - \boldsymbol{\omega}\boldsymbol{\mu}_Z\boldsymbol{\mu}_Z^t\boldsymbol{\omega} \tag{3}$$

$$\gamma_{1,\ell} = \frac{E[(X_\ell - E(X_\ell))^3]}{Var(X_\ell)^{3/2}} = \frac{4-\pi}{2}\frac{\mu_{Z;\ell}^3}{(1-\mu_{Z;\ell}^2)^{3/2}} \ , \ \ell = 1,\ldots,q \tag{4}$$

where $\boldsymbol{\mu}_Z$ is a vector of expected values for standard Skew-Normal variables, which are defined by

$$\boldsymbol{\mu}_Z = \sqrt{\frac{2}{\pi}}\frac{\boldsymbol{\omega}^{-1}\boldsymbol{\Omega}\boldsymbol{\omega}^{-1}\boldsymbol{\alpha}}{\sqrt{1+\boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}\boldsymbol{\Omega}\boldsymbol{\omega}^{-1}\boldsymbol{\alpha}}}$$

As an alternative to the Gaussian model, it may be considered that (C, R^*) follows jointly a $2p$-multivariate Skew-Normal distribution, for which the different alternative configurations of the $\boldsymbol{\Sigma}$ matrix may be assumed. Given (3), a null covariance $\boldsymbol{\Sigma}(j,j')$ implies that $\boldsymbol{\Omega}(j,j') = \boldsymbol{\Omega}(j,j)^{\frac{1}{2}}\boldsymbol{\mu}_{Z_j}\boldsymbol{\Omega}(j',j')^{\frac{1}{2}}\boldsymbol{\mu}_{Z_{j'}}$ or, equivalently, $\boldsymbol{\Omega}(j,j) = \dfrac{2}{\pi}\dfrac{1}{1+\boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}\boldsymbol{\Omega}\boldsymbol{\omega}^{-1}\boldsymbol{\alpha}}\boldsymbol{\Omega}_j^t\boldsymbol{\omega}^{-1}\boldsymbol{\alpha}\boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}\boldsymbol{\Omega}_{j'}$ where $\boldsymbol{\Omega}_j$ denotes the j^{th} column of $\boldsymbol{\Omega}$. This defines non-linear relations between the parameters in $\boldsymbol{\Omega}$ and $\boldsymbol{\alpha}$.

Maximum likelihood estimation

As discussed in the previous subsection, Brito and Duarte Silva (2012) consider as possible models for interval-valued data, eight possible combinations of two multivariate distributions (Gaussian or Skew-Normal) with four covariance configurations. Given an observed data set, the choice among these models may be based on their maximised likelihood using usual information criteria such as the Bayesian Information Criterion (BIC) (Schwarz, 1978), the Akaike Information Criterion (AIC) (Akaike, 1974), or pairwise likelihood ratio tests. In this subsection we will present the details of the respective maximum likelihood estimation.

Gaussian model

Let $\boldsymbol{X_i} = \left[\boldsymbol{C_i}^t, \boldsymbol{R_i^*}^t\right]^t$ be the $2p$-dimensional column vector comprising all the MidPoints and Log-Ranges for unit s_i, $\bar{\boldsymbol{X}}$ be sample mean of the $\boldsymbol{X_i}$'s and $\boldsymbol{E} = \sum_{i=1}^{n}(\boldsymbol{X_i}-\bar{\boldsymbol{X}})(\boldsymbol{X_i}-\bar{\boldsymbol{X}})^{\boldsymbol{t}}$. For all configurations, the log-likelihood can be written as

$$\ln L(\boldsymbol{\mu},\boldsymbol{\Sigma}) = l = -np\ \ln(2\pi) - \frac{n}{2}\ \ln\det\boldsymbol{\Sigma} - \frac{1}{2}\ \mathrm{Tr}(\boldsymbol{E}\boldsymbol{\Sigma}^{-1}) - \frac{n}{2}\,(\bar{\boldsymbol{X}}-\boldsymbol{\mu})^t\,\boldsymbol{\Sigma}^{-1}\,(\bar{\boldsymbol{X}}-\boldsymbol{\mu}) \tag{5}$$

Under the unrestricted configuration C1, the maximum likelihood estimators of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are obviously the classical ones, $\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{X}}$ and $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\boldsymbol{E}$. In the restricted configurations C2 to C4, the maximum of (5) can be obtained by separately maximising with respect to each block of $\boldsymbol{\Sigma}$, and the estimators are obtained from the non-restricted estimators simply replacing the null parameters in $\boldsymbol{\Sigma}$ by zeros (see Brito and Duarte Silva (2012)).

Skew-Normal model

Azzalini and Capitanio (see, e.g., Azzalini and Capitanio (1999); Azzalini (2005)) have obtained the log-likelihood of a q-dimensional Skew-Normal distribution as

$$\ln L(\boldsymbol{\xi},\boldsymbol{\Omega},\boldsymbol{\alpha}) = l = \text{ constant } -\frac{1}{2}n\ln\det\boldsymbol{\Omega} - \frac{n}{2}\,\mathrm{Tr}(\boldsymbol{\Omega}^{-1}\mathbf{V}) + \sum_i \zeta_0(\boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}(\boldsymbol{X_i}-\boldsymbol{\xi})) \tag{6}$$

where $\boldsymbol{V} = n^{-1}\sum_i(\boldsymbol{X_i}-\boldsymbol{\xi})(\boldsymbol{X_i}-\boldsymbol{\xi})^t$ and $\zeta_0(v) = \ln(2\Phi(v))$. The maximisation of (6) is performed in two steps by defining a new parameter, $\boldsymbol{\eta} = \boldsymbol{\alpha}^t\boldsymbol{\omega}^{-1}$, and separating the maximisation on $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ from the maximisation on $\boldsymbol{\Omega}$ given $\boldsymbol{\xi}$, which has the analytical solution $\boldsymbol{\Omega} = \boldsymbol{V}$.

The optimal likelihood solution for the Skew-Normal model with restricted configurations may not be obtained by simply replacing corresponding entries in the appropriate matrices, because of the non-linear relations between the parameters in $\boldsymbol{\Omega}$ and $\boldsymbol{\alpha}$. For the Skew-Normal model with restricted configurations, we rely on a centred parametrisation (Valle and Azzalini, 2008), which employs directly the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and γ_1 given by (2), (3) and (4), respectively. The log-likelihood is maximised with respect to $\boldsymbol{\mu}$, γ_1 and the free elements in $\boldsymbol{\Sigma}$. This optimisation must be done numerically; see Subsection **Implementation** of Section **Package** for the details of the implementation adopted in package **MAINT.Data**.

Robust estimation and outlier detection

Multivariate datasets often include data units that deviate from the main pattern, usually called *outliers*, which may strongly influence the maximum likelihood estimators, leading to the need of alternative (robust) estimators. In the context of interval-valued data this problem has been addressed in Duarte Silva et al. (2018).

There is an extensive literature on robust estimation of location and scatter parameters. Trimmed likelihood estimators (Hadi and Luceño, 1997) are based on a sample subset, keeping only the h units that contribute most to the likelihood function. For multivariate Gaussian data, this approach is equivalent to the well-known Minimum Covariance Determinant (MCD) method (Rousseeuw, 1984, 1985) which consists in using the sample subset that minimises the determinant of the covariance matrix estimate (Hadi and Luceño, 1997). Since finding the true MCD is an NP-hard problem, when n is not small, a good approximation based on a computationally fast algorithm is usually employed (Rousseeuw and Van Driessen, 1999).

Outlier detection usually relies on Mahalanobis distances, flagging units as outliers if their distances from an appropriate estimate of the multivariate mean $\boldsymbol{m}$ is above a chosen quantile of an appropriate distribution. (Squared) Mahalanobis distances are defined as $D^2_{\boldsymbol{m},\boldsymbol{C}}(i) = (\boldsymbol{X_i}-\boldsymbol{m})^t\boldsymbol{C}^{-1}(\boldsymbol{X_i}-\boldsymbol{m})$ where $\boldsymbol{C}$ is an estimate of the covariance matrix. Traditionally, a Chi-square approximation is used for the distribution of MCD-robust squared Mahalanobis distances; however, Cerioli (2010) proposed finite sample approximations with better properties for small and even moderately large sample sizes.

Moreover, more efficient one-step re-weighted MCD estimators are often used (Hubert et al., 2008). These are obtained by giving null weight only to the units for which the raw squared robust Mahalanobis distance exceeds a high threshold value, e.g., the 97.5% quantile of the classical Chi-square or, alternatively, of the scaled-F approximation (Hardin and Rocke, 2005). Furthermore, the resulting covariance estimators are usually multiplied by consistency and bias correction factors (see Pison et al. (2002)).

In practice, one needs to specify the number h of data points to be initially used. Two common choices are to fix this number around 50%n maximising the breakdown point, or around 75%n for larger efficiency (Hubert et al., 2008). Recently, in the context of interval data outlier identification, Duarte Silva et al. (2018) proposed a two-step approach where the outlier detection procedure is first run to get an estimate of the outlier proportion and in a second step the procedure is repeated fixing the trimming parameter at the value obtained in the first step.

The trimmed Maximum Likelihood approach described above has been adapted to the problem of robust parameter estimation for the Gaussian models proposed for interval-valued data. For all considered covariance configurations, the trimmed log-likelihood can be written as

$$\ln\, TL(\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{h}{2}\left(2p\;\ln(2\pi)+\ln\;\det\boldsymbol{\Sigma}+\mathrm{Tr}(\tilde{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1})+(\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu})^{t}\,\boldsymbol{\Sigma}^{-1}\,(\tilde{\boldsymbol{\mu}}-\boldsymbol{\mu})\right) \tag{7}$$

where h is the number of observations kept in the trimmed sample, and

$\tilde{\boldsymbol{\mu}} = \frac{1}{h}\sum_{i=1}^{h}\boldsymbol{X_i}$, $\tilde{\boldsymbol{\Sigma}} = \frac{1}{h}\sum_{i=1}^{h}(\boldsymbol{X_i}-\tilde{\boldsymbol{\mu}})(\boldsymbol{X_i}-\tilde{\boldsymbol{\mu}})^t$ are the trimmed mean and trimmed sample covariance, respectively.

In the case of a restricted covariance matrix, the block diagonal structure always implies that trimmed likelihood maximisation is equivalent to the minimisation of the determinant of the restricted trimmed sample covariance matrix.

The one-step re-weighted bias-corrected estimators are given by

$$\hat{\boldsymbol{\mu}}_1 = \frac{\sum_{i=1}^{n} w_i\boldsymbol{X_i}}{h_1} \tag{8}$$

$$\hat{\boldsymbol{\Sigma}}_1 = \frac{l_{h_1}\,c_1\sum_{i=1}^{n} w_i(\boldsymbol{X_i}-\hat{\boldsymbol{\mu}}_1)(\boldsymbol{X_i}-\hat{\boldsymbol{\mu}}_1)^t}{h_1} \tag{9}$$

$$h_1=\sum_{i=1}^{n} w_i \qquad w_i=\begin{cases}1, & \text{if}\quad l_h c\, D_{\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}}}(i)\le\sqrt{Q_{0.975}}\\ 0, & otherwise\end{cases}$$

where $Q_{0.975}$ is the 97.5% quantile of the $D^2_{\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}}}$ distribution. In **MAINT.Data** this distribution is approximated by a Chi-square distribution with $2p$ degrees of freedom or by a (scaled) F distribution as proposed by Hardin and Rocke (2005).

In expression (9), l_h and l_{h_1} are consistency correction factors, whereas c and c_1 are finite-sample bias-correction factors - for more details see Duarte Silva et al. (2018).

These estimates may then be used for outlier detection in an interval-valued dataset. For that purpose, the robust squared Mahalanobis distance for unit i, based on $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\Sigma}}_1$, is compared with the chosen upper quantile of either the χ^2_{2p} distribution or using the approximations (see Cerioli (2010)):

$$D^2_{\hat{\boldsymbol{\mu}}_1,\hat{\boldsymbol{\Sigma}}_1} \sim \frac{(h_1-1)^2}{h_1} Beta\left(p,\frac{h_1-2p-1}{2}\right),\quad if\;\; w_i=1 \tag{10}$$

$$D^2_{\hat{\boldsymbol{\mu}}_1,\hat{\boldsymbol{\Sigma}}_1} \sim \frac{h_1+1}{h_1}\frac{(h_1-1)2p}{h_1-2p}F\left(2p,h_1-2p\right),\quad if\;\; w_i=0 \tag{11}$$

Multivariate analysis

Analysis of Variance

The models presented above for interval-valued variables allow addressing (M)ANOVA problems with interval data - see Brito and Duarte Silva (2012). Since each interval-valued variable Y_j is modelled by $\left[C_{ij},R^*_{ij}\right]$, an analysis of variance of Y_j is accomplished by a two-dimensional MANOVA.

Assume a one-way design, with a single factor with k levels, and let n_ℓ be the number of units in group ℓ. Let $X_{ij}=\left[C_{ij},R^*_{ij}\right]^t$ be the 2-dimensional column vector with the MidPoint and Log-Range of variable Y_j for unit s_i, and let $\mu_{\bullet j\ell}$ be the population means of the X_j's in group ℓ. In this case, the null hypothesis states that all $\mu_{\bullet j\ell}$ are equal across groups. In all cases, for both models and all covariance configurations, we follow a likelihood ratio approach.

In the Gaussian model, the usual likelihood ratio statistic λ can be computed in a straightforward manner. Under the unrestricted case C1, this statistic is obviously equal to the classical one; in the restricted covariance cases, its value may be obtained replacing the null entries corresponding to each configuration in the sum of squares and cross-products MANOVA matrices (see e.g. Huberty and Olejnik (2006) for the definition of those matrices). For the Skew-Normal model, given there is no closed form for the maximum likelihood estimates, the value of λ must be obtained by numerical optimisation (see Subsection **Implementation** of Section **Package**).

As usual, under the null hypothesis, $-2\ln\lambda$, follows asymptotically a Chi-square distribution. For small samples a permutation test may be used to approximate the distribution of this test statistic.

A simultaneous analysis of all the Y's interval-valued variables may be accomplished by a $2p$-dimensional MANOVA, following the same procedure.

Discriminant Analysis

The classical decision theoretic approach to Discriminant Analysis assumes that a given vector of attributes follows some known distribution and derives an optimal classification rule that minimises either the misclassification probability or the expected value of the misclassification cost. Parametric discriminant analysis of interval-value data based on the models above has been investigated in Duarte Silva and Brito (2015).

In a problem with k groups, $\Gamma_\ell, \ell = 1, \ldots, k$, denote the *a priori* group membership probabilities by π_ℓ and the within group probability or density function by $f_\ell(\mathbf{x})$, where $\mathbf{x}$ are attribute vectors. Under the assumption that misclassification cost are equal across groups, the optimal rule assigns a unit to the group Γ_ℓ for which $\pi_\ell f_\ell(\mathbf{x})$ is maximal (see, e.g. McLachlan (1992)); in practice the unknown parameters in these rules must be estimated from observations with known group membership.

When $f_\ell(.)$ is a Gaussian density, and the covariance matrices are equal across groups, the approach described above leads to a linear classification rule, whereas when covariance matrices differ from group to group, a quadratic classification rule is obtained.

Consider the Gaussian model for interval data. For each covariance configuration, an estimate of the optimum classification rule can be obtained by direct generalisation of the classical linear and quadratic discriminant classification rules, leading to group assignments defined by, respectively,

$$\Gamma = \underset{\ell}{\operatorname{argmax}}(\hat{\boldsymbol{\mu}}_\ell^t \hat{\boldsymbol{\Sigma}}^{-1}\mathbf{x} - \frac{1}{2}\hat{\boldsymbol{\mu}}_\ell^t \hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_\ell + \ln\ \hat{\pi}_\ell) \tag{12}$$

$$\Gamma = \underset{\ell}{\operatorname{argmax}}(-\frac{1}{2}\mathbf{x}^t\hat{\boldsymbol{\Sigma}}_\ell^{-1}\mathbf{x} + \hat{\boldsymbol{\mu}}_\ell^t\hat{\boldsymbol{\Sigma}}_\ell^{-1}\mathbf{x} + \ln\ \hat{\pi}_\ell - \frac{1}{2}(\ln\det\hat{\boldsymbol{\Sigma}}_\ell + \hat{\boldsymbol{\mu}}_\ell^t\hat{\boldsymbol{\Sigma}}_\ell^{-1}\hat{\boldsymbol{\mu}}_\ell)) \tag{13}$$

where $\hat{\boldsymbol{\mu}}_\ell, \hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\Sigma}}_\ell$ and $\hat{\pi}_\ell$ are appropriate estimates of $\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_\ell$ and π_ℓ for the corresponding cases.

In **MAINT.Data**, all mean and covariance estimates in (12) and (13) may be obtained by either classical maximum likelihood or the robust trimmed maximum likelihood approach (see Section **Robust estimation and outlier detection**).

We note that for the restricted configurations C2, C3 and C4, $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\Sigma}}_\ell$ are obtained from the corresponding unrestricted estimates replacing all the null covariances by zeros.

For the Skew-Normal case, we consider a Location Model in which the groups differ only in terms of the location parameter $\boldsymbol{\xi}$, and a General Model, where the groups differ in terms of all parameters. The corresponding classification rules are, respectively,

$$\Gamma = \underset{\ell}{\operatorname{argmax}}(\hat{\boldsymbol{\xi}}_\ell^t\,\hat{\boldsymbol{\Omega}}^{-1}\mathbf{x} - \frac{1}{2}\hat{\boldsymbol{\xi}}_\ell^t\,\hat{\boldsymbol{\Omega}}^{-1}\hat{\boldsymbol{\xi}}_\ell + \ln\ \hat{\pi}_\ell + \zeta_0(\hat{\boldsymbol{\alpha}}^t\hat{\boldsymbol{\omega}}^{-1}(\mathbf{x} - \hat{\boldsymbol{\xi}}_\ell))) \tag{14}$$

$$\Gamma = \underset{\ell}{\operatorname{argmax}}(-\frac{1}{2}\mathbf{x}^t\hat{\boldsymbol{\Omega}}_\ell^{-1}\mathbf{x} + \hat{\boldsymbol{\xi}}_\ell^t\,\hat{\boldsymbol{\Omega}}_\ell^{-1}\mathbf{x} + \ln\ \hat{\pi}_\ell - \tag{15}$$
$$\frac{1}{2}(\ln\ \det\hat{\boldsymbol{\Omega}}_\ell + \hat{\boldsymbol{\xi}}_\ell^t\,\hat{\boldsymbol{\Omega}}_\ell^{-1}\hat{\boldsymbol{\xi}}_\ell) + \zeta_0(\hat{\boldsymbol{\alpha}}_\ell^t\,\hat{\boldsymbol{\omega}}_\ell^{-1}(\mathbf{x} - \hat{\boldsymbol{\xi}}_\ell)))$$

where $\hat{\boldsymbol{\xi}}_\ell, \hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\Omega}}_\ell, \hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\alpha}}_\ell$ are estimates of location, scale, association and shape parameters (see Azzalini and Capitanio (1999)), $\hat{\boldsymbol{\omega}}$ and $\hat{\boldsymbol{\omega}}_\ell$ are the square-roots of the diagonal elements of the matrices $\hat{\boldsymbol{\Omega}}$ and $\hat{\boldsymbol{\Omega}}_\ell$, respectively, and $\zeta_0(v) = \ln(2\Phi(v))$. In **MAINT.Data** these are all maximum likelihood estimates.

Model-based Clustering

Model-based Clustering considers the data as arising from a distribution that is a mixture of two or more components (Banfield and Raftery, 1993; Fraley and Raftery, 2002; McLachlan and Peel, 2000). Each component, that can be thought as a cluster, is characterised by a conditional density/mass function and has an associated probability or "weight". When the conditional probability is specified as the multivariate Gaussian density, the probability model for clustering will be a finite mixture of multivariate Normals (known as the Gaussian mixture model).

The problem consists in estimating the model parameters for each component, as well as the membership (posterior) probabilities of each unit. To this purpose, the Expectation-Maximisation (EM) algorithm (Dempster et al., 1977) is commonly used. The method alternates between an expectation (E) step, which computes the expectation of the log-likelihood at the current parameter estimates, and a maximisation (M) step, which estimates parameters maximising the expected log-likelihood found in the E step.

Model-based Clustering of interval data has been addressed in Brito et al. (2015), by considering the Gaussian parametrisation described above (see Section **Models and estimation**). For that purpose, the EM algorithm has been adapted to the likelihood maximisation in our models, for the different covariance configurations.

The finite mixture model with k components for $2p$-dimensional data vector $\boldsymbol{x}$ is defined by

$$f(\boldsymbol{x};\boldsymbol{\varphi}) = \sum_{\ell=1}^{k} \tau_\ell f_\ell(\boldsymbol{x};\boldsymbol{\theta_\ell}) \tag{16}$$

where all $\tau_\ell > 0$ and $\tau_1 + \ldots + \tau_k = 1$; $\boldsymbol{\theta_\ell}$ denotes parameters of the conditional distribution of component ℓ.

Here the conditional distribution is given by $N(\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)$; maximum likelihood parameter estimation involves the maximisation of the log-likelihood function:

$$\ln L(\boldsymbol{\varphi};\mathbf{x}) = \sum_{i=1}^{n} \ln f(\mathbf{x}_i;\boldsymbol{\varphi}) \tag{17}$$

where $\boldsymbol{\varphi} = (\tau_1, \ldots, \tau_k, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_k})$.

In Model-based clustering of interval data, $\boldsymbol{X_i} = \left[\boldsymbol{C_i^t}, \boldsymbol{R_i^*}^t\right]^t$ is defined as the $2p$-dimensional vector comprising all the MidPoints and Log-Ranges for s_i, and the "complete" data are considered to be $(\boldsymbol{x_i}, \boldsymbol{z_i})$, where $\boldsymbol{z_i} = (z_{i1}, \ldots, z_{ik})$ is assumed as the "missing" data, with $z_{i\ell} = 1$ if $s_i \in$ component ℓ and $z_{i\ell} = 0$ otherwise. In the unrestricted case, the M-step formulas for $\hat{\boldsymbol{\Sigma}}$, $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\ell}}$ are the classical ones; for the restricted configurations $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\ell}}, \ell = 1, \ldots, k$ are obtained maximising the likelihood for each block separately (see Brito and Duarte Silva (2012)).

For the selection of the appropriate model and the number of components k, we use the Bayesian Information Criterion (BIC).

Package

Design

MAINT.Data is built around S4 classes and methods, the most important being the IData class and classes derived from the virtual IdtE (IData Estimates) classes. Further specialised classes used to store the results of various multivariate analysis (*e.g.* Model-based Clustering, MANOVA and Discriminant Analysis) are also available. Figure 1 shows common interactions between different objects of **MAINT.Data** classes.

We note that in addition to the flow shown in Figure 1, objects containing the results of Discriminant Analysis of Interval Data may also be obtained from appropriate objects of class `IdtMANOVA`, or directly from the combination of objects of class IData with a grouping factor.

Class `Idata`, which is used to store datasets of interval-valued variables, is the central class in the **MAINT.Data** package. Its design aims at replicating the functionalities of classical data frames as smoothly as possible. As seen in Figure 1, objects of class `IData` may be created in one of two alternative ways: (i) directly from data frames containing either lower and upper bounds or MidPoints and Log-Ranges, using the creator function `IData`; (ii) by aggregation of a data frame of the microdata by a given aggregating factor and criterion (*e.g.* min-max or a given pair of quantiles), using the function `AgrMcDt`.

The creator function `IData` takes five arguments as input. The first one, named *Data* refers to a data frame or matrix containing either the lower and upper bounds or the MidPoints and Log-Ranges of the observed intervals, where each row corresponds to a different unit. Then, *Seq* is a string which describes the sequence of the data for each unit, namely, lower and upper bounds variable by variable (*"LbUb_VarbVar"*, default), MidPoints and Log-Ranges variable by variable (*"MidPLogR_VarbVar"*), all

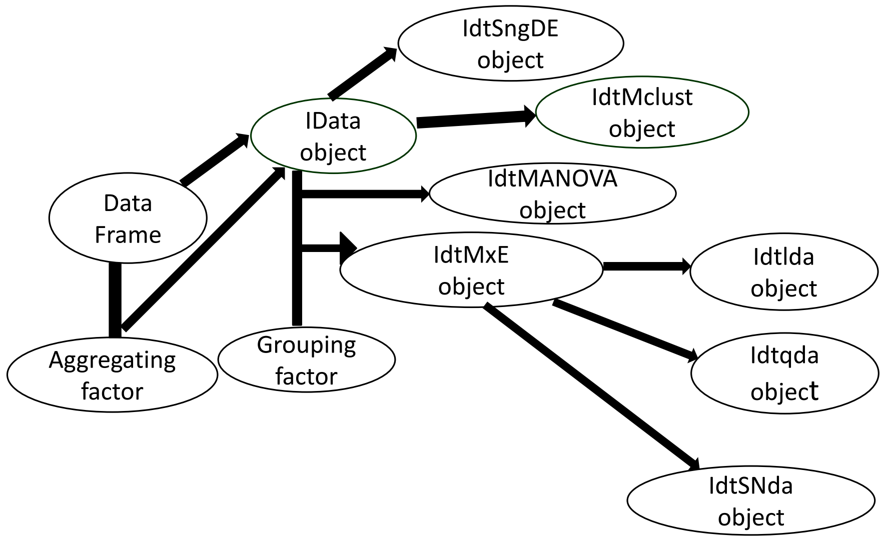


Figure 1: Typical flow of a MAINT.Data application.

lower bounds followed by all upper bounds (*"AllLb_AllUb"*), or all MidPoints followed by all Log-Ranges (*"AllMidP_AllLogR"*). The third and fourth arguments, named *VarNames* and *ObsNames*, allow the user to specify the variables' and units' names, respectively. Finally, the last argument *NbMicroUnits* provides the number of micro observations corresponding to each unit, when available. A typical call of this function would be `ExampleIdt <-IData(dataDF,VarNames=c(``Var1'',``Var2''))` (no names for the units, number of micro observations corresponding to each unit not available).

Function `AgrMcDt` has three arguments. The first one, *MicDtDF* indicates a data frame with the microdata. The second argument, *agrby* refers to a factor with the categories according to which the microdata should be aggregated. The last argument *agrcrt* specifies whether aggregation is done with the minimum and maximum observed values, or else based on user-defined quantiles. An example is shown in Section **Applications**.

A UML diagram of class `Idata` is shown in Figure 2. As seen here, class `Idata` implements specialised versions of standard R methods such as `summary`, `print`, `nrow` and `ncol`, `rownames` and `colnames`, `rbind`, `cbind` and `plot`. Special care has been taken in the development of indexing operators and of a specialised `cbind` method, so that they work as smoothly as with data frames, but treating each column of `Idata` as one interval-valued variable.

The remaining `Idata` methods perform parameter estimation and/or multivariate analysis leading to objects of class `IdtE` (parameter estimation), `IdtMANOVA` (Multivariate Analysis of Variance), `Idtda` (Discriminant Analysis), or `IdtMclust` (Model-based Clustering). All these methods include a `Covcase` argument used to specify the covariance configurations assumed, which by default compares the BIC of the results for all four configurations, and select the one with the lowest BIC value.

The `IdtE` class is an abstract (virtual) class used to store parameter estimates of the models assumed for interval-valued variables. As shown in Figure 3 there are currently eight such specialisations, depending on the model assumed and type of estimation performed. The names of these classes always start with the letters *Idt* followed by *Sng* or *Mx* (estimates of parameters of a single or several distributions), *ND*, *SND* or *NandSND* (Gaussian, SkewNormal or both Gaussian and SkewNormal distributions), and end with *E* or *RE* (Maximum Likelihood or Robust estimates).

As shown in Figure 4 the same reasoning applies to classes derived from the virtual class `IdtMANOVA`. However, in this case, only Maximum Likelihood estimation has been considered and the specialisations distinguish classical MANOVA (class `IdtClMANOVA`), heterocedastic MANOVA based on Gaussian

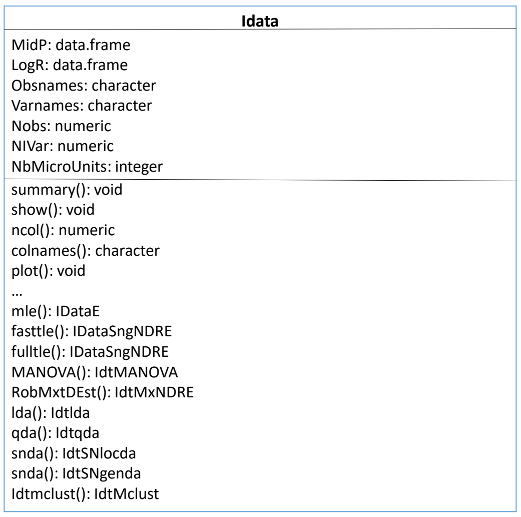


Figure 2: IData class.

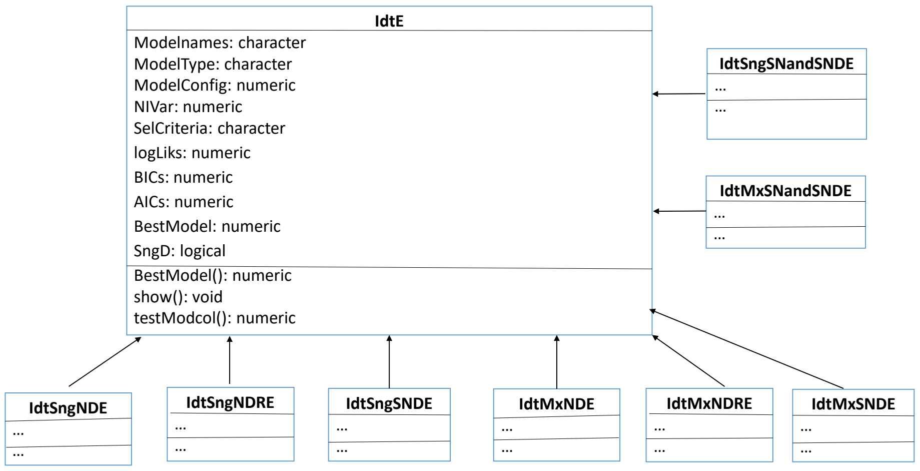


Figure 3: IdtE classes.

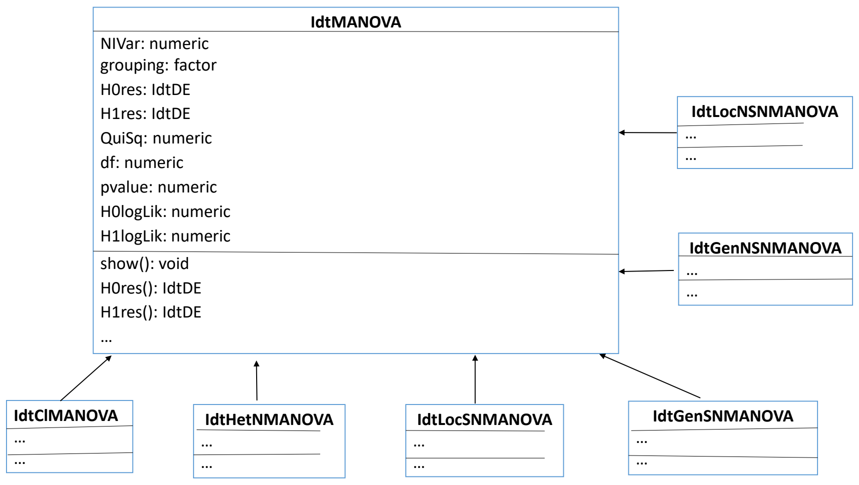


Figure 4: IdtMANOVA classes.

distributions (`IdtHetNMANOVA`), Skew-Normal based MANOVA assuming that groups may differ only in location (`IdtLocNMANOVA`) or on all parameters (`IdtGenNMANOVA`), and analyses that consider both Gaussian and SkewNormal assumptions (`IdtLocNSNMANOVA` and `IdtGenNSNMANOVA`).

Maximum likelihood estimation is performed by the `mle` method, which has six arguments. The first one, *Idt* refers to an IData object representing interval-valued units. The second argument, *Model* indicates the joint distribution assumed for the MidPoint and LogRanges; alternatives are "Normal" for Gaussian (default), "SKNormal" for Skew-Normal and "NrmandSKN" for both Gaussian and Skew-Normal distributions. The next argument, *CovCase* indicates the configurations of the variance-covariance matrix to be used (default: 1:4). The fourth argument, *SelCrit* indicates the model selection criterion, BIC (default) or AIC. The argument *kmax* specifies a tolerance criterion to identify singular correlation matrices. Finally, *OptCntrl* provides a list of optional control parameters to be passed to the optimization routine.

Robust estimation is usually performed by the `fasttle` method. Note that for small datasets, the `fulltle` method may be used, whose arguments are common to `fasttle`. The first three arguments of `fasttle` are the same as for the `mle` method. Arguments *alpha* and *getalpha* specify how the trimming proportion is chosen. Other important arguments are the following: *use.correction* indicates whether to use finite sample correction factors, default is TRUE. *rawMD2Dist* provides the assumed reference distribution of the raw MCD squared distances used to find the cutoffs defining the observations kept in one-step reweighted MCD estimates; alternatives are "ChiSq" for the usual Chi-square (default), "HardRockeAsF" and "HardRockeAdjF", respectively asymptotic and adjusted scaled F distributions proposed by Hardin and Rocke (2005). *MD2Dist* - assumed reference distribution used to find cutoffs defining the observations assumed as outliers; alternatives are "ChiS" and "CerioliBetaF", respectivelly for the usual Chi-square, and the Beta and F distributions proposed by Cerioli (2010). *reweighted* indicates whether a (Re)weighted estimate of the covariance matrix should be used in the computation of the trimmed likelihood or just a "raw" covariance estimate; default is TRUE. Argument *outlin* specifies the type of outliers to be considered, alternatives are "MidPandLogR" if outliers may be present in both MidPoints and LogRanges, "MidP" if outliers are only present in MidPoints, or "LogR" if outliers are only present in LogRanges.

Method `MANOVA` applies multivariate analysis of variance. The arguments *Idt*, *Model*, *CovCase*, *SelCrit*, *k2max* and *OptCritl* are identical to the corresponding ones of method `mle`. Argument *grouping* indicates the factor whose levels are the different groups. *MxT* indicates the type of mixing distributions to be considered: "Hom" (homoscedastic) or "Het" (heteroscedastic) for Gaussian models, "Loc" (location model) or "Gen" (general model) for Skew-Normal models (see Section **Discriminant Analysis** above). *CVtol* provides a tolerance value to identify almost constant variables within groups.

To perform discriminant analysis, three methods may be applied, namely, `lda` (linear discriminant analysis), `qda` (quadratic discriminant analysis) and `snda` (skew-normal based discriminant analysis). In all these methods, the first argument *x* denotes an IData object representing the interval-valued units, or else an object of class `IdtMANOVA`. Arguments *grouping*, *CVtol*, *CovCase*, *SelCrit* and *k2max* are identical to the corresponding ones of method `MANOVA`. The argument *prior* is used to specify the prior probabilities of group membership, by default they are fixed at the training set corresponding proportions. In method snda, argument *MxT* indicates the type of mixing distributions to be considered: "Loc" (location model, default) or "Gen" (general model).

Method Idtmclust performs model-based clustering based on finite mixtures of Gaussian distributions. Arguments *Idt*, *CovCase*, and *SelCrit* are identical to the corresponding ones in the previous methods. The argument *G* provides the number of clusters (segments) of the mixture, by default it is set as 1:9. *MxT* indicates the type of mixing distributions to be considered, "Hom" (homoscedastic, default), "Het" (heteroscedastic), or "HomandHet" (both). Finally, the argument *control* provides a list of control parameters for the EM algorithm.

Implementation

The implementation of the Idata class, as well as maximum likelihood estimation and multivariate methods based on the Gaussian distribution, is relatively straightforward. As shown in Figure 2, the internal structure of the Idata class consists of two data frames, containing MidPoints and Log-Ranges, respectively, a couple of auxiliary constants and vector strings, and the integer vector NbMicroUnits which stores the number of microunits aggregated to form each interval-valued unit, when known. Therefore, Idata objects require roughly twice the memory space used by traditional data frames. The Idata slots may be retrieved by the accessor methods MidPoints, LogRanges, rownames, colnames, nrow, and ncol.

The structure of the classes derived from the virtual IdtE class (see Figure 3) depends on the type of model specified and estimation performed. In addition to the common slots of the IdtE class, these classes include vector and/or matrix slots with estimates that are constant across all covariance configurations, and a list slot named *ConvConfCases* in which each component contains estimates obtained under the assumption of a particular configuration. We note that, although the estimates corresponding to one single configuration are displayed and used in further analysis, all estimates resulting from the configurations specified by the argument *CovCase* are stored, and available to the user. The same logic applies to analyses that consider more than one model, with the results for all models being stored, but only one displayed by summary and print methods.

The maximum likelihood estimation and multivariate analysis based on the Gaussian distribution do not entail any particular difficulties, usually involving well known formulae and the replacement of some values by zero according to the covariance configuration assumed. Covariance matrices of Gaussian estimators are also computed in a straightforward manner and passed, if so requested, to the appropriate stdEr and vcov methods.

Maximum likelihood estimation of Skew-Normal parameters requires the numerical optimisation of the non-convex function (6). As this function often has many different local optima, **MAINT.Data** adopts a repeated local search strategy, calling a given local optimiser from different starting points. This is implemented in the auxiliary function RepLOptim that works as described below.

First, a local optimiser is called from an initial starting point leading to a local optimum. Then, this optimum is modified by a random perturbation, and the modified optimum is used as the starting point of a new call to the local optimiser. This process is repeated until several (default: 50) consecutive calls to the optimiser fail to improve the current best solution, or a limit (default: 250) on the total number of local optima, is reached. This limit, the maximum number of non-improving consecutive local optimisations, and several other control options, are set by default to reasonable values, but can be modified by the components of a list supplied as the value of the argument *control*. The same applies to methods (such as mle or MANOVA or snda) that internally call RepLOptim, using in this case a list supplied to their *Optcontrol* argument.

The default local optimiser of RepLOptim is the nlminb PORT function (Gay, 1990). However, in the case of maximum likelihood estimation of Skew-Normal parameters with unrestricted covariance configuration (C1), **MAINT.Data** overrides this default with the msn.mle function of Azzalini's **sn** package (Azzalini, 2021). For the remaining configurations, the local optimisation relies on a quasi-Newton optimiser (by default nlminb) using the analytical gradient of the centred Skew-Normal parametrisation derived by Valle and Azzalini (2008). In order to improve computational efficiency, the computation of the log-likelihood (6) and of its gradient was coded in C++, taking advantage of the numerical functions and classes provided in the **Rcpp** (Eddelbuettel and François, 2011) and **RcppArmadillo** (François et al., 2021) packages. We note that global optimality cannot be ensured and even with this strategy sometimes **MAINT.Data** identifies different local optima in different runs.

Once the optimisation of the log-likelihood (6) is completed, **MAINT.Data** approximates the covariance of the estimators using the evaluation of the expected Fisher information matrix implemented in the **sn** package. This approximation may fail if either the expected information matrix is ill-conditioned or the parameter estimates fall on the frontier of their domain. In such cases, posterior calls to the stdEr or vcov methods will result in appropriate warning messages.

The robust estimation of Gaussian model parameters by the trimmed maximum likelihood principle is implemented in the fulltle and fasttle methods. Method fulltle makes a full combinatorial search for the Trimmed Maximum Likelihood estimates, and should only be used when the number of units is relatively small (say, not much larger than 40). Method fasttle adapts the fast algorithm of Rousseeuw and Driessen (Rousseeuw and Van Driessen, 1999). Both methods were coded in C++, using functions and classes from **Rcpp** and **RcppArmadillo**. Furthermore, the methods RobMxtDESt, Roblda and Robqda call fasttle or fulltle in order to get robust estimates in different groups that may be used for robust discriminant analysis.

The interface of the **MAINT.Data** robust methods and classes is partially based on the framework developed in the popular **rrcov** package (Todorov and Filzmoser, 2009). In particular, the control options for the estimation algorithm used in the fasttle, RobMxtDESt, Roblda and Robqda methods can be provided by an argument of class RobEstControl which inherits and extends the class CovControl of the package **rrcov**. This way, algorithmic options may be specified in a uniform and familiar manner. The additional slots of class RobEstControl specify new options, such as indicators of the distributions assumed for the robust Mahalanobis distances, the nature of the outliers (only in MidPoints, only in Log-Ranges or (default) both in MidPoints and Log-Ranges), whether a two-step procedure should be used to find trimming parameters, and other choices that are available in **MAINT.Data** but not in **rrcov**.

The MANOVA methods available in **MAINT.Data** are always based on the maximum likelihood principle. By default, the Chi-square distribution is used for the test statistic. However, for small samples, a permutation test has been implemented in the auxiliary function MANOVAPermTest (see Seber (2009)).

The design and interface of class IdtMclust is modelled after class Mclust of the **mclust** package (Scrucca et al., 2016). As a result, the IdtMclust print and summary methods with their default argument values, display only a very general description of the clustering results. A characterization of the obtained clusters, and the partition itself, may be inspected by changing the summary arguments *parameters*, and *classification* from *FALSE* to *TRUE*. A difference between the mclust and IdtMclust classes lies in that in the former case detailed clustering results can only be retrieved directly from the Mclust slots while IdtMclust provides accessor methods such as parameters, pro, mean, var, cor and classification to retrieve these results. The EM algorithm used in IdtMclust is implemented in C++, using facilities of the **Rcpp** and **RcppArmadillo** packages.

Application I: flights dataset

To illustrate the modelling and methods presented above, we use the flights dataset from the R data package **nycflights13**, available at *CRAN*, which contains on-time data for all flights that departed New York city in 2013. The original microdata consists of 336776 flights characterized by nineteen variables.

From this data, we created a data frame named *FlightsDF*, after removing all rows with missing data, and with six columns corresponding to the following descriptive variables at microdata level: Month, Carrier (16 different carriers), Departure delay (min), Arrival delay (min), Air time(min), and Distance (miles).

We consider as units of interest classes formed by crossing Month with Carrier, leading to 185 units (note that not all the 192 possible combinations are present in the microdata). Therefore, we created a factor, named *FlightsUnits*, defining the class each individual case belongs to.

The command

```
R> FlightsIdt <- AgrMcDt(FlightsDF,FlightsUnits)
```

creates an interval data object *FlightsIdt*, where the values of the numerical variables Departure delay (DD), Arrival delay (AD), Air time (AT) and Distance (DT) are aggregated in the form of intervals for each unit. Leaving the aggregation argument agrcrt at its "minmax" default, the lower and upper bounds of the obtained intervals are the minimum and maximum values observed in the microdata, respectively.

However, we prefer to use the robust aggregation alternative, by filtering out the 5% lowest and highest values for each variable; in this case the aggregation argument specifies the chosen pair of quantiles:

```
R> FlightsIdt <- AgrMcDt(FlightsDF,FlightsUnits,agrcrt=c(0.05,0.95))
```

We note that the 43 units for which, for any variable, the lower and the upper bound are equal (degenerate interval) are eliminated, so that the final interval dataset has 142 units.

Table 3 shows a few rows of the resulting interval data table. The full interval dataset is available on **MAINT.Data**.

	Departure delay	Arrival delay	Air time	Distance
Jan-9E	$[-10, 120]$	$[-32, 116]$	$[31, 176]$	$[94, 1029]$
Jan-AA	$[-9, 65]$	$[-31, 59]$	$[115, 354]$	$[733, 2475]$
...	...	...	...	...
Dec-YV	$[-10.9, 68.2]$	$[-33.5, 66]$	$[39.4, 102.8]$	$[96, 544]$

Table 3: Flights interval data - partial view.

Figure 5 illustrates the two alternative outputs - (a)-crosses, (b)-rectangles - of the method `plot`, resulting respectively from the commands

```
R> plot(FlightsIdt[,"distance"],FlightsIdt[,"arr_delay"],
       cex.main=3, cex.lab=1.9, cex.axis=2)
R> plot(FlightsIdt[,"distance"],FlightsIdt[,"arr_delay"],type="rectangles",
       cex.main=3, cex.lab=1.9, cex.axis=2)
```

showing the intervals corresponding to the 142 units in two different forms for variables Distance and Arrival delay.

We note that the graphical arguments of traditional R plots are also available in **MAINT.Data** `plot` methods. In this example, the default graphical settings were adequate for online display, but resulted in too small axis and legends, when the resulting graphs were exported to an external text file. Therefore, we used the `cex.main`, `cex.lab`, and `cex.axis` traditional R plot arguments, to improve their readability. This particular example worked well on a PC under Linux, but since graphical characteristics are machine and operation system dependent, other argument values may be required in different computer environments.

Figure 6 plots the MidPoints versus the Log-Ranges for variable Arrival delay, resulting from the command

```
R> plot(MidPoints(FlightsIdt)[,"arr_delay.MidP"],
       LogRanges(FlightsIdt)[,"arr_delay.LogR"],
       xlab="Mid Points",ylab="Log Ranges",
       main="Mid Points vs. Log Ranges for Arrival delays",
       cex.main=2, cex.lab=1.5, cex.axis=1.5)
```

We observe a strong positive correlation between the MidPoints and the Log-Ranges of the Arrival delay, which is not uncommon for interval-valued variables.

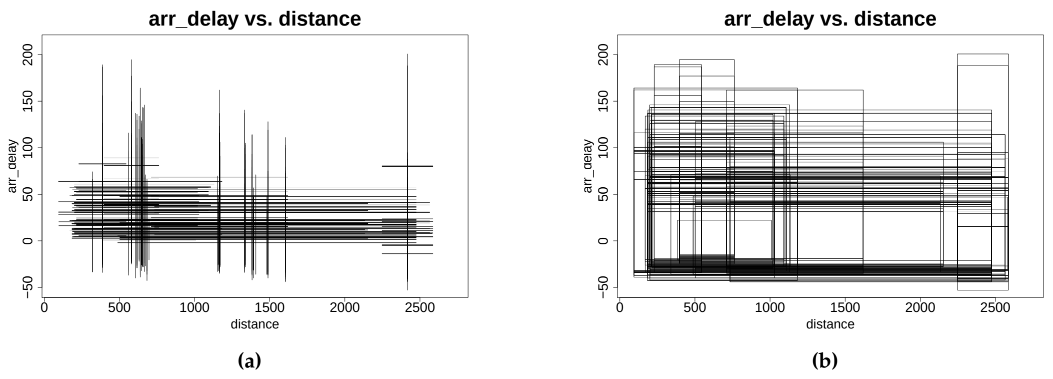


Figure 5: Interval representation of the 142 flights units – Distance versus Arrival delay.

Modelling of flights interval data

The following statistical analyses of the data will be directed towards the methods proposed earlier. Accordingly, a first analysis relates to statistically characterize the input variables and possible rela-

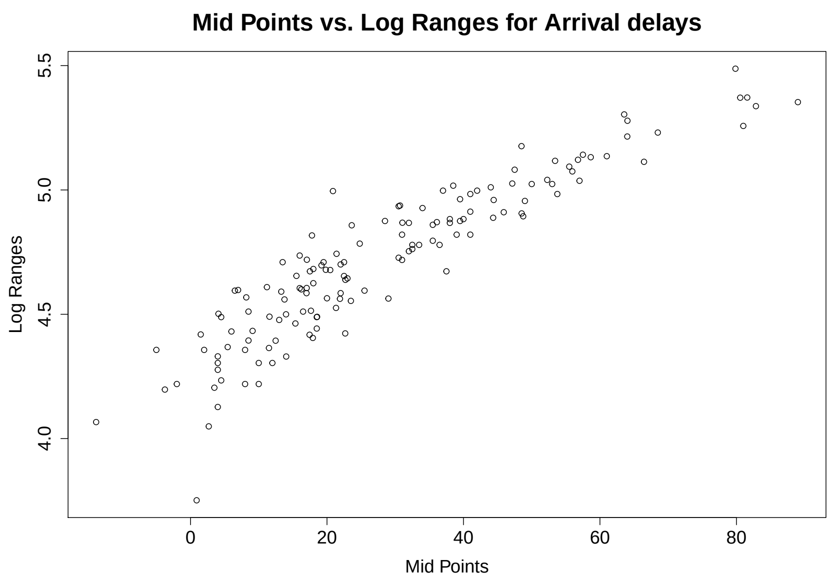


Figure 6: Midpoints VS Log-Ranges of variable Arrival delay for the 142 flights units.

tionships between them. Then the interest is in possible outliers in the interval data. Are there specific carrier/month combinations which are atypical for the observed features? The identified outliers will be excluded from the data for the subsequent analyses, which could be affected by data outliers. The data are split into two groups, the mainline carriers and the regional carriers. Do these groups differ for the considered variables (MANOVA)? Is it possible to distinguish the observations of the two groups from each other (discriminant analysis)? Are there even more subgroups in the multivariate data, and how can those be characterized (cluster analysis)?

We start by adjusting the Gaussian and Skew-Normal models for all four considered covariance configurations using the commands

```
R> Flightsmle <-mle(FlightsIdt,Model="NrmandSKN")
R> summary(Flightsmle)
```

which produce the output

```
Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2
-2278.626  -3249.162  -2547.369  -3564.026  -2207.444  -3183.179
SNModCovC3 SNModCovC4
-2491.175  -3498.258
Bayesian (Schwartz) Information Criteria:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2
4775.309   6597.440   5233.501   7207.346   4672.591   6505.121
SNModCovC3 SNModCovC4
5160.760   7115.455
Selected model:
[1] "SNModCovC1"
```

We recall that for the Skew-Normal model only local optima are identified, so that in different runs slightly different solutions may be obtained.

Among the eight models $\times$ configurations, the BIC recommends the Skew-Normal model with covariance configuration C1. The likelihood ratio tests between pairs of models may be performed by command `testMod(Flightsmle)`. In this case, for any reasonable significance level, these tests suggest also the Skew-Normal model with covariance configuration C1.

The estimates of mean, standard deviation and skewness coefficient vectors and the variance-covariance matrix may be extracted by the usual `coef()` method. Alternatively, the standard methods `mean()`, `sd()`, `var()` and `cor()` may also be used. Furthermore, standard errors and variances and covariances of the estimates may be obtained, as usual, by the methods `stdEr()` and `vcov()`.

The estimates for mean values, standard deviations and skewness coefficients are:

$$\begin{array}{rcccccccccc}
 & & \multicolumn{4}{c}{C} & \multicolumn{4}{c}{R^*} & \\
 & & DD & AD & AT & DT & DD & AD & AT & DT & \\
\hat{\mu}^t = & (& 38.14 & 27.98 & 179.70 & 1244.63 & 4.45 & 4.72 & 4.89 & 6.84 &) \\
\hat{\sigma}^t = & (& 19.30 & 20.68 & 61.89 & 476.72 & 0.42 & 0.32 & 0.59 & 0.66 &) \\
\hat{\gamma}^t = & (& 0.000 & -0.001 & 0.271 & 0.247 & 0.000 & 0.000 & -0.046 & -0.064 &)
\end{array}$$

The estimate of the correlation matrix is :

$$\hat{\mathbf{R}} = \left(\begin{array}{cc|cccc|cccc}
 & & \multicolumn{4}{c|}{C} & \multicolumn{4}{c}{R^*} \\
 & & DD & AD & AT & DT & DD & AD & AT & DT \\
 & DD & 1.00 & 0.96 & -0.32 & -0.30 & 0.96 & 0.96 & -0.32 & -0.31 \\
\mathrm{C} & AD & & 1.00 & -0.40 & -0.39 & 0.92 & 0.92 & -0.33 & -0.33 \\
 & AT & & & 1.00 & 0.99 & -0.37 & -0.27 & 0.47 & 0.43 \\
 & DT & & & & 1.00 & -0.35 & -0.24 & 0.49 & 0.45 \\
\hline
 & DD & & & & & 1.00 & 0.97 & -0.31 & -0.29 \\
\mathrm{R}^* & AD & & & & & & 1.00 & -0.25 & -0.23 \\
 & AT & & & & & & & 1.00 & 0.99 \\
 & DT & & & & & & & & 1.00
\end{array}\right)$$

We observe that MidPoints are positively correlated with the corresponding Log-Ranges, with strong correlations for the delay variables and moderate correlations for Distance and Air Time. The MidPoints of Departure delay and Arrival delay on the one hand, and Air time and Distance, on the other hand, have, as expected, strong correlations; the corresponding Log-Ranges also present high correlations. The observed correlation values explain the choice of the unrestricted covariance configuration C1.

Robust estimation results are obtained by the commands:

```
R> Flightstle  <-fasttle(FlightsIdt)
R> summary(Flightstle)
```

which produce the output

```
Log likelihoods:
NModCovC1 NModCovC2 NModCovC3 NModCovC4
-1416.930 -2069.002 -1720.921 -2490.111
Bayesian (Schwartz) Information Criteria:
NModCovC1 NModCovC2 NModCovC3 NModCovC4
3040.279  4231.831  3573.200  5055.283
Selected model:
[1] "NModCovC1"
```

The estimates for mean values and standard deviations are now:

$$\begin{array}{rcccccccccc}
 & & \multicolumn{4}{c}{C} & \multicolumn{4}{c}{R^*} & \\
 & & DD & AD & AT & DT & DD & AD & AT & DT & \\
\hat{\mu}^t = & (& 35.99 & 26.51 & 159.09 & 1096.43 & 4.454 & 4.70 & 5.22 & 7.23 &) \\
\hat{\sigma}^t = & (& 18.02 & 19.65 & 58.82 & 459.60 & 0.42 & 0.32 & 0.56 & 0.59 &)
\end{array}$$

To identify outliers, we employ the default options for the robust methods and functions implemented in **MAINT.Data**, namely a cut-off based on the Chi-square distribution, and a trimming parameter based on a two step procedure using 75% of the sample in the first step.

The following command allows obtaining the list of units identified as outliers:

```
R> Flights_Otl <- getIdtOutl(FlightsIdt,Flightstle)
```

```
R> print(Flights_Otl)
```

which returns

```
Jan-FL  Jan-VX  Feb-FL  Feb-VX  Mar-FL  Mar-VX  Apr-FL  Apr-VX  Apr-YV
6      10      17      21      28      32      39      43      45
May-FL  May-VX  Jun-9E  Jun-FL  Jun-VX  Jun-WN  Jun-YV  Jul-9E  Jul-FL
51      55      58      63      67      68      69      70      75
Jul-VX  Aug-FL  Aug-VX  Aug-YV  Sep-VX  Sep-YV  Oct-FL  Oct-VX  Nov-FL
79      87      91      93     103     105     111     115     123
Nov-OO  Nov-VX  Nov-YV  Dec-FL  Dec-VX
125     128     130     136     140
```

From this list it is visible that FL (AirTran Airways) is an outlier for almost all months. When inspecting the aggregated data, it can be seen that the upper bound of the Distance is clearly lower than for the other airlines, and consequently also the upper bound of Air time. The contrary happens for airline VX (Virgin America), which is an outlier for all months. Note, however, that outlyingness can also be caused by a different multivariate behaviour of an observation.

Figure 7 shows the values of robust Mahalanobis distances (to the mean) for all 142 units, the horizontal line indicates the 97.5% quantile of the respective Chi-square distribution; it is obtained by the command

```
R> plot(Flights_Otl, cex.main=2, cex.lab=1.5, cex.axis=0.3)
```

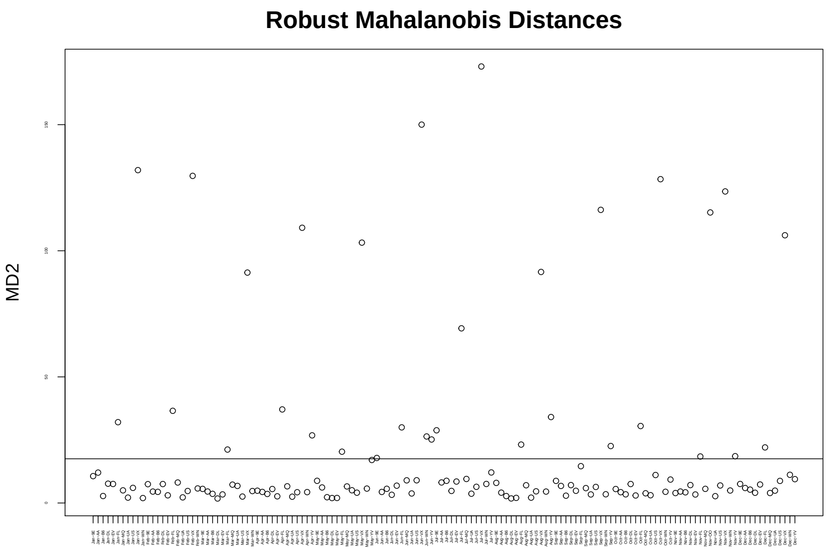


Figure 7: Values of robust Mahalanobis distances (to the mean) for the 142 flights units.

MANOVA

We now consider a partition of the 110 regular (i.e. not flagged as outliers) flights units into two groups according to whether the airline is a mainline or a regional one; the 5 regional airlines are Endeavor Air Inc. (9E), ExpressJet Airlines Inc. (EV), Envoy Air (MQ), SkyWest Airlines Inc. (OO), and Mesa Airlines Inc. (YV).

MANOVA analysis was performed for all flights units, considering this two group decomposition. We compared both a Gaussian and a Skew-Normal model, with all four covariance configurations (default), with a homoscedastic setup (default), and using the BIC (default) as comparison criterion. For that purpose we used the commands

```
R> out1<-Flights_Otl@outliers
R> carr <- substring(rownames(FlightsIdt[-out1,]),5,6)
R> carr_class <- factor(ifelse(carr=="9E"|carr=="EV"|carr=="MQ"|
 carr=="OO"|carr=="YV","REG","MAIN"))
```

```
R> MANOVAres <- MANOVA(FlightsIdt[-out1,],carr_class,Model="NrmandSKN")
R> summary(MANOVAres)
```

leading to the output

```
Null Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-1437.000  -2094.997  -1743.016  -2521.166  -1397.125  -2060.841  -1735.816  -2444.528
Full Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-1318.227  -1874.335  -1566.672  -2184.943  -1259.869  -1821.942  -1536.026  -2085.836
Full Model Bayesian (Schwartz) Information Criteria:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
2880.879   3880.282   3302.561   4482.697   2801.767   3813.102   3278.874   4322.088
Selected Model:
[1] "SNModCovC1"

Chi-squared statistic: 274.5117
degrees of freedom: 8
p-value: 1.082712e-54
```

The Skew-Normal model with variance-covariance configuration C1 is selected (on the basis of BIC values) as the best model in this case, results indicate that the two carrier groups are indeed different for the considered variables.

These results were to be expected given that regional carriers tend to fly short distances and therefore with shorter air times, than mainlines.

We then proceeded to investigate whether the two carrier groups are different when it comes to each of the delay variables, using the commands

```
R> MANOVA_Dep_delay_res <-  MANOVA(FlightsIdt[-out1,"dep_delay"],carr_class,Model="NrmandSKN")
R> summary(MANOVA_Dep_delay_res)
R> MANOVA_Arr_delay_res <-  MANOVA(FlightsIdt[-out1,"arr_delay"],carr_class,Model="NrmandSKN")
R> summary(MANOVA_Arr_delay_res)
```

that produced the outputs

```
Null Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-313.5688         NA         NA  -492.4602  -287.8342         NA         NA  -473.1451
Full Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-296.6978         NA         NA  -469.0213  -272.3174         NA         NA  -449.6624
Full Model Bayesian (Schwartz) Information Criteria:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
626.2990         NA         NA   966.2455   586.9391         NA         NA   936.9286
Selected Model:
[1] "SNModCovC1"

Chi-squared statistic: 31.03359
degrees of freedom: 2
p-value: 1.824489e-07

Null Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-366.8737         NA         NA  -473.4313  -360.3910         NA         NA  -457.5188
Full Model Log likelihoods:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
-357.0406         NA         NA  -456.5808  -351.1153         NA         NA  -441.4999
Full Model Bayesian (Schwartz) Information Criteria:
NModCovC1  NModCovC2  NModCovC3  NModCovC4 SNModCovC1 SNModCovC2 SNModCovC3 SNModCovC4
746.9845         NA         NA   941.3644   744.5350         NA         NA   920.6036
Selected Model:
[1] "SNModCovC1"

Chi-squared statistic: 18.55139
degrees of freedom: 2
p-value: 9.367357e-05
```

The results show that the two carrier groups present different patterns for both departure and arrival delays.

Discriminant Analysis

We consider again the 110 regular flights units grouped in two carrier classes, as in Section **Analysis of Variance**.

The function `DACrossVal` estimates error rates by *c-fold* cross-validation or by leave-one-out. Its main arguments are: (i) the data object, (ii) the grouping factor, (iii) a function with the training algorithm (e.g. lda, qda, snda, see Subsection **Design** of Section **Package**), (iv) the number of cross-validation folds (default: 10), (v) the number of replications (default: 20), (vi) a boolean flag indicating whether the folds should be stratified according to the original class proportions (default), or randomly generated from the whole training sample, ignoring class membership, and (vii) a boolean flag (false by default) stating if the leave-one-out method should be used instead of *c-fold* cross-validation.

Different discriminant methods were compared by leave-one-out cross-validation: Linear and Quadratic Discriminant Analysis for the Gaussian model, and both the Location and the General models of Skew-Normal discriminant analysis. In each case the variance-covariance configuration (see Table 2) was chosen by minimising the value of BIC. The global errors, which are also provided, are computed as a weighted average of the estimated class specific errors.

The code below computes and displays these estimates.

```
R>  DACrossVal(FlightsIdt[-out1,],carr_class,TrainAlg=lda,loo=TRUE)
R>  DACrossVal(FlightsIdt[-out1,],carr_class,TrainAlg=qda,loo=TRUE)
R>  DACrossVal(FlightsIdt[-out1,],carr_class,TrainAlg=snda,loo=TRUE)
R>  DACrossVal(FlightsIdt[-out1,],carr_class,TrainAlg=snda,Mxt="Gen",loo=TRUE)
```

We note that while the first two commands are executed quite fast (a few seconds), the last two (for the Skew-Normal model) typically need several hours, given that a computationally heavy Skew-Normal estimation is repeated many times.

These commands lead to the output below. Note that when `snda` is used without any additional arguments, the default location model is assumed.

```
Error rate estimates of algorithm lda
MAIN         REG         Global
0.013888889 0.000000000 0.009090909


Error rate estimates of algorithm qda
MAIN        REG        Global
0.00000000 0.05263158 0.01818182


Error rate estimates of algorithm snda
MAIN        REG        Global
0.08333333 0.07894737 0.08181818

Error rate estimates of algorithm snda with argument Mxt=Gen
MAIN        REG        Global
0.05555556 0.18421053 0.10000000
```

These results suggest that `lda` performs better than the other alternatives in the problem at hand. The predicted classes using the `lda` method may be obtained, as usual, by the commands

```
R> ldares <- lda(FlightsIdt[-out1,],carr_class)
R> ldapred <- predict(ldares,FlightsIdt[-out1,])
R> print(ldapred$class)
```

We observed that all but one units are correctly classified, namely carrier FL in September, the only FL unit which was not flagged as an outlier.

Clustering the Flights units

Model-based Clustering described in Section **Model-based Clustering** was applied to the Flights dataset, without the identified outliers, to identify up to 16 components. This is accomplished by the command:

```
R> mclust_res <- Idtmclust(FlightsIdt[-out1,],1:16,Mxt="HomandHet")
```

where again, by default, the recommended solution is selected by the BIC. The corresponding values may be graphically compared using the command

```
R> plotInfCrt(mclust_res, cex.lab=1.5, outlegsize=10, outlegdisp=0.25)
```

which provided the graphic in Figure 8 and the output below. A homocedastic nine component model has been selected with an unrestricted covariance configuration C1.

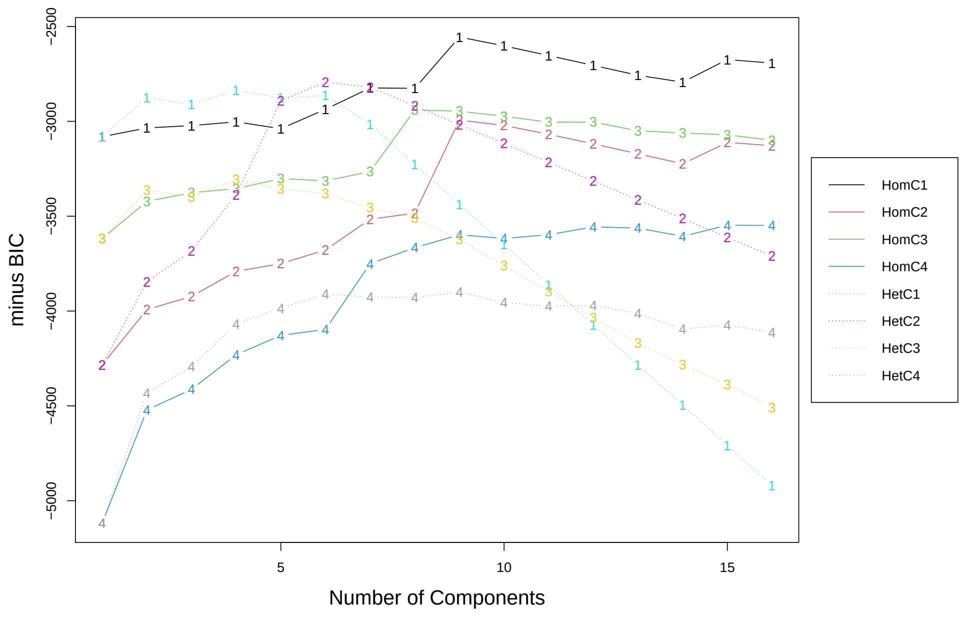


Figure 8: BIC values for different models and number of components.

```
Best BIC values:
HomG9C1    HomG10C1    HomG11C1    HomG15C1    HomG16C1
BIC          2555.407    2602.228    2653.101    2674.13    2692.822
BIC diff            0    46.82063    97.69343    118.7225    137.4153
```

The value returned by `Idtmclust` is an object of class `IdtMclust` with the same structure, and similar methods, of the corresponding `Mclust` class of package **mclust** (Scrucca et al., 2016). In particular, the classification results may be inspected by the command:

```
R> summary(mclust_res,classification=TRUE)
```

which returns

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------
Homoscedastic C1 model with 9 components
```

```
log.likelihood NObs      BIC
-1005.076      110       2555.407

Clustering table:
CP1 CP2 CP3 CP4 CP5 CP6 CP7 CP8 CP9
12  12  14  12  10  12  24   9   5

Classification:
Jan-9E Jan-AA Jan-B6 Jan-DL Jan-EV Jan-MQ Jan-UA Jan-US Jan-WN Feb-9E
"CP5"  "CP2"  "CP7"  "CP4"  "CP5"  "CP6"  "CP7"  "CP3"  "CP3"  "CP1"
Feb-AA Feb-B6 Feb-DL Feb-EV Feb-MQ Feb-UA Feb-US Feb-WN Mar-9E Mar-AA
"CP2"  "CP7"  "CP4"  "CP5"  "CP6"  "CP7"  "CP3"  "CP3"  "CP1"  "CP2"
Mar-B6 Mar-DL Mar-EV Mar-MQ Mar-UA Mar-US Mar-WN Apr-9E Apr-AA Apr-B6
"CP7"  "CP4"  "CP5"  "CP6"  "CP7"  "CP3"  "CP8"  "CP1"  "CP2"  "CP7"
Apr-DL Apr-EV Apr-MQ Apr-UA Apr-US Apr-WN May-9E May-AA May-B6 May-DL
"CP4"  "CP5"  "CP6"  "CP7"  "CP3"  "CP8"  "CP5"  "CP2"  "CP7"  "CP4"
May-EV May-MQ May-UA May-US May-WN May-YV Jun-AA Jun-B6 Jun-DL Jun-EV
"CP1"  "CP6"  "CP7"  "CP3"  "CP8"  "CP9"  "CP2"  "CP7"  "CP4"  "CP5"
Jun-MQ Jun-UA Jun-US Jul-AA Jul-B6 Jul-DL Jul-EV Jul-MQ Jul-UA Jul-US
"CP6"  "CP7"  "CP3"  "CP2"  "CP7"  "CP4"  "CP5"  "CP6"  "CP7"  "CP3"
Jul-WN Jul-YV Aug-9E Aug-AA Aug-B6 Aug-DL Aug-EV Aug-MQ Aug-UA Aug-US
"CP8"  "CP9"  "CP1"  "CP2"  "CP7"  "CP4"  "CP1"  "CP6"  "CP7"  "CP3"
Aug-WN Sep-9E Sep-AA Sep-B6 Sep-DL Sep-EV Sep-FL Sep-MQ Sep-UA Sep-US
"CP8"  "CP1"  "CP2"  "CP7"  "CP4"  "CP1"  "CP9"  "CP6"  "CP7"  "CP3"
Sep-WN Oct-9E Oct-AA Oct-B6 Oct-DL Oct-EV Oct-MQ Oct-UA Oct-US Oct-WN
"CP8"  "CP1"  "CP2"  "CP7"  "CP4"  "CP1"  "CP6"  "CP7"  "CP3"  "CP8"
Oct-YV Nov-9E Nov-AA Nov-B6 Nov-DL Nov-EV Nov-MQ Nov-UA Nov-US Nov-WN
"CP9"  "CP1"  "CP2"  "CP7"  "CP4"  "CP1"  "CP6"  "CP7"  "CP3"  "CP8"
Dec-9E Dec-AA Dec-B6 Dec-DL Dec-EV Dec-MQ Dec-UA Dec-US Dec-WN Dec-YV
"CP5"  "CP2"  "CP7"  "CP4"  "CP5"  "CP6"  "CP7"  "CP3"  "CP8"  "CP9"
```

We observe that units corresponding to the same carrier tend to cluster together, for instance, component 2 gather all AA units, component 4 gathers DL units and component 6 the MQ units.

In order to have a better description of the obtained partition, we then print the corresponding mixing probabilities and the component-wise mean vectors.

```
R> print(pro(mclust_res), digits=3)
R> print(mean(mclust_res),digits=3)
```

obtaining

```
CP1     CP2     CP3     CP4     CP5     CP6     CP7     CP8     CP9
0.10684 0.10909 0.12727 0.10752 0.09316 0.10909 0.21976 0.08182 0.04545

                  CP1     CP2     CP3     CP4    CP5    CP6     CP7     CP8    CP9
dep_delay.MidP  40.26   31.02   21.06   27.39  59.83  35.15   34.85   43.51  44.85
arr_delay.MidP  25.84   18.28   15.44   16.45  51.08  31.89   25.00   32.63  34.63
air_time.MidP   99.13  223.40  163.22  208.73 101.86  98.97  193.37  171.20  72.08
distance.MidP  640.43 1604.00 1163.86 1481.65 632.53 616.50 1360.07 1165.50 411.80
dep_delay.LogR   4.59    4.35    4.07    4.19   4.92   4.48    4.41    4.55   4.68
arr_delay.LogR   4.76    4.68    4.42    4.59   5.05   4.71    4.69    4.80   4.80
air_time.LogR    4.83    5.47    5.55    5.56   4.86   4.63    5.71    4.91   3.97
distance.LogR    6.86    7.46    7.58    7.59   6.81   6.69    7.75    6.81   5.85
```

These mean vectors can be compared by a parallel coordinate plot, using the pcoordplot method as follows

```
R> pcoordplot( mclust_res, cex.main=2, cex.lab=2,
     legendpar=list(cex.main=2.5, cex.lab=2.5) )
```

which produces the graph shown in Figure 9.

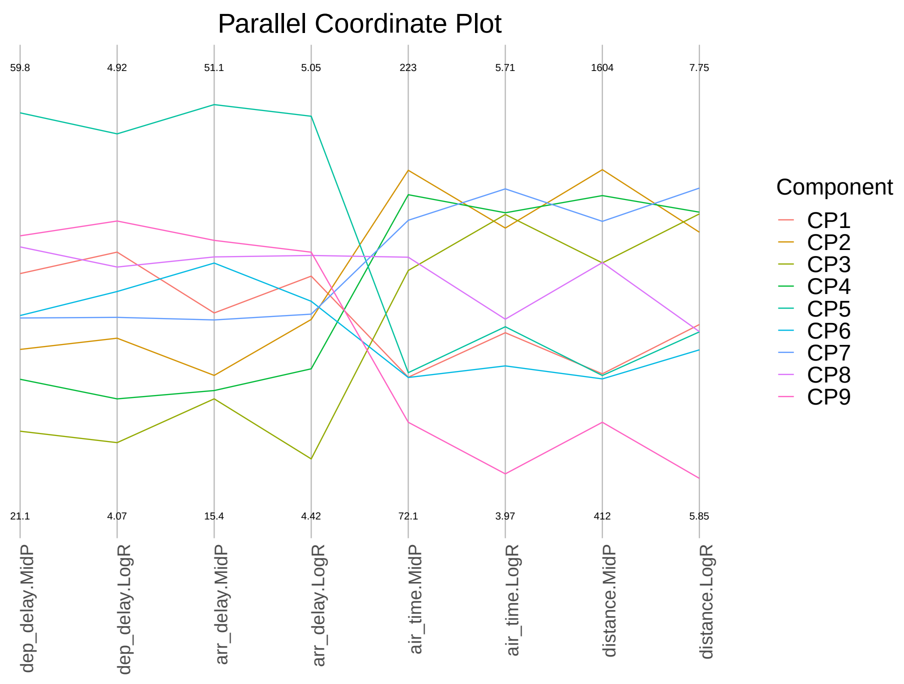


Figure 9: Parallel coordinate plot of best clustering solution, with nine components.

We observe that component 5 is mainly characterised by the largest delays both at departures and arrivals, also displaying their highest variability. Component 3, on the other hand, presents the lowest delays, with lowest variability, and concerns long flights. Component 9 corresponds to the shortest flights, with also low variability of distance and airtime. Components 4 and 7 present similar patterns in the distance and airtime variables, although component 4 displays slightly larger midpoints while component 7 has a higher variability; in terms of delays, we observe in component 7 higher values together with a more important variability.

From Figure 8 we observe that the best heterocedastic model corresponds to configuration C2 and identifies six components.

The corresponding mean vectors may again be displayed by a parallel coordinate plot, using the `pcoordplot` method, now indicating the solution of interest:

```
R> pcoordplot(mclust_res, model="HetG6C2", cex.main=2, cex.lab=2,
    legendpar=list(cex.main=2.5, cex.lab=2.5) )
```

leading to the graph shown in Figure 10.

Application II: diamonds dataset

This second example explores the diamonds dataset (from the R package **tidyverse** available at CRAN). The original microdata consists of 53940 diamonds characterised by ten variables. Descriptive variables are: *carat* (weight of the diamond), *x* (length in mm), y (width in mm), and *z* (depth in mm). All rows with missing data or null values in at least one of these variables were removed. Because the distribution of these variables is positive skewed, they were log-transformed (natural logarithm).

The units of analysis were defined by the variables: *cut* (quality of the cut: *Fair*, *Good*, *Very Good*, *Premium*, *Ideal*), *color* (diamond color, with seven levels: *J* (worst) to *D* (best)), and clarity (measurement of how clear the diamond is, with eight levels: *I1* (worst), *SI2*, *SI1*, *VS2*,

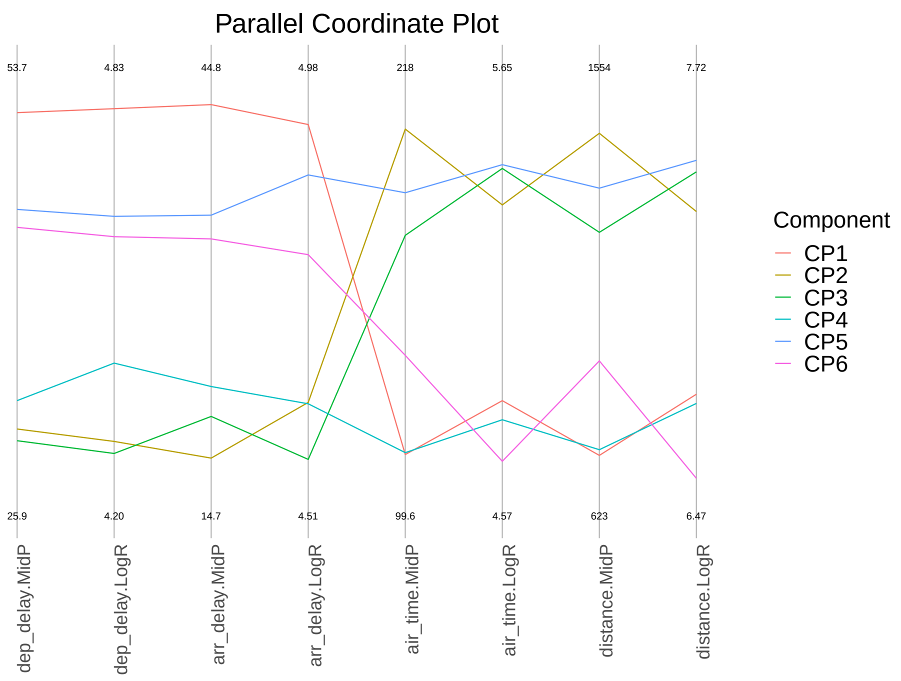


Figure 10: Parallel coordinate plot of best heterocedastic solution, with six components.

VS1, *VVS2*, *VVS1*, *IF* (best)), totalising 271 units (out of the 280, four were not present in the data, and five were degenerated). The variable *DiamondsUnits* defines these combinations.

The commands

```
R> library(tidyverse)

R> valid_diamonds  <- diamonds %>%
R> filter(carat !=0, x != 0, y != 0, z != 0) %>%
R> drop_na() %>% mutate(logcarat = log(carat),
                        logx     = log(x),
                        logy     = log(y),
                        logz     = log(z))

R> DiamondsUnits <- factor( paste(
    valid_diamonds$cut,valid_diamonds$color,valid_diamonds$clarity, sep="-"
  ) )

R> DiamondsIdt <- AgrMcDt(valid_diamonds[,c("logcarat","logx","logy","logz")],
                          agrby=DiamondsUnits)
```

do the initial data processing and create the interval data object *DiamondsIdt* using the default option (min-max). In the application we do not filter out potential outliers as we want to use the finite mixture model to detect them. Indeed, outliers can be seen as an unstructured component of the mixture model (Aitkin and Wilson, 1980).

From the estimation of mixtures from one to eight components, we have

```
R> Diamd_mclust_res <- Idtmclust(DiamondsIdt,1:8,Mxt="HomandHet")
R> plotInfCrt(Diamd_mclust_res, cex.lab=1.5, outlegsize=10, outlegdisp=0.25)

Best BIC values:
              HetG3C3     HetG4C3     HetG3C1     HetG5C3     HetG6C3
BIC         -7818.702   -7789.946   -7752.533   -7702.916   -7637.323
BIC diff            0    28.75589    66.16948    115.7865    181.3797
```

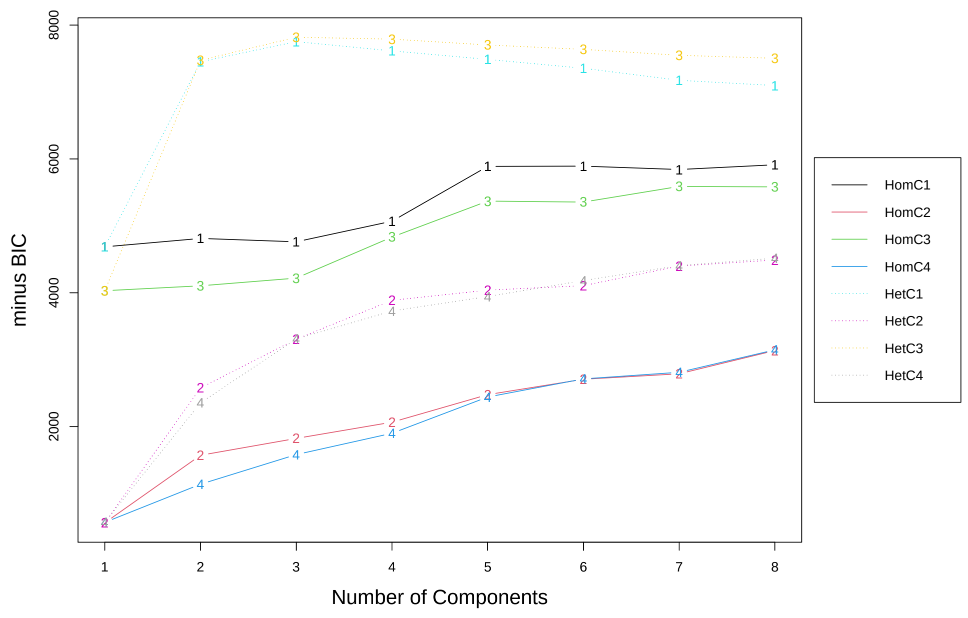


Figure 11: BIC values for different models and number of components.

Selection based on BIC recommends configuration 3 with three components (see Figure 11). Thus, the best solution is a heteroscedastic solution in which centers are not correlated with ranges.

```
R> summary(Diamd_mclust_res)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------
Heteroscedastic C3 model with 3 components
 log.likelihood NObs       BIC
       4150.242  271 -7818.702

Clustering table:
CP1 CP2 CP3
 11 176  84

R> print(pro(Diamd_mclust_res), digits=3)
   CP1    CP2    CP3
0.0407 0.6328 0.3265
```

We conclude that the size of *CP1* is 0.0407 and contains eleven observations (hard classification), i.e., it is an outlier or niche group. Whenever a population/sample is well represented by the Normal distribution, a single component (or point of support) is enough to model its density. Often, however, distributions tend to have skewness and kurtosis that are far from the multivariate Normal distribution. In that case, Gaussian mixture models (GMM) have been used to estimate densities as an alternative to nonparametric or semi-parametric Kernels (Scott, 2015). Indeed, this application of finite mixtures is more general than model-based clustering as the latter tends to be specific to the correspondence between modes and clusters or groups. In this example, the departure from normality (skewness and kurtosis) is modeled using two additional components.

```
R> plot(MidPoints(DiamondsIdt)[,"logz.MidP"],
     LogRanges(DiamondsIdt) [,"logz.LogR"],
     xlab="Mid Points",ylab="Log Ranges",
     main="Mid Points vs. Log Ranges for logdepth in mm",
     col = ifelse(Diamd_mclust_res@classification == 'CP1',1,
                  ifelse(mclust_res@classification == 'CP2',2,7)),
                  pch = 19, cex.lab=1.5)
```

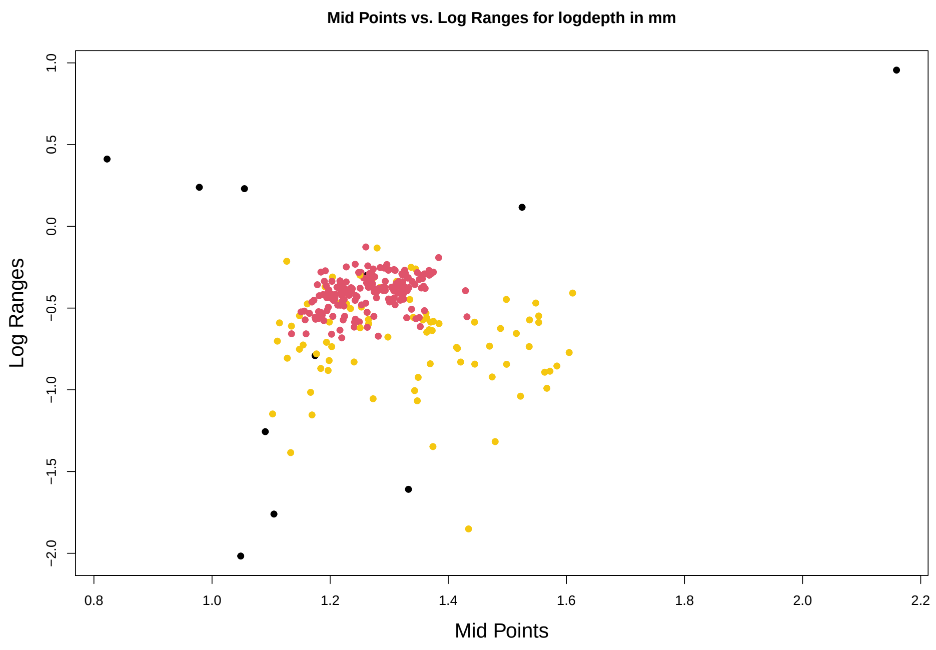


Figure 12: Representation of classified units (*logdepth*).

Figure 12 obtained from the code above illustrates the approximation of the density of the data by the finite mixture. The core group is well defined by the multivariate normal distribution. As the result of skewness, a second group is added. And then, finally, the heavy (multidimensional) "tail" is given by the outlier component (black dots) with its large estimated variances and co-variances.

Summary

The MAINT.Data R package implements models and methods for the analysis of interval-valued data, relying on multivariate Normal or Skew-Normal distributions for the MidPoints and Log-Ranges of the interval-valued variables. Implemented in the S4 framework, it introduces a data class for representing interval data and functions and methods for parametric modelling and analysis.

The available tools for interval variable management include interval-data versions of most of the standard R methods such as print and summary, index and subseting, and plot. Moreover, functions for aggregating microdata into interval data objects are also provided. The multivariate methodologies available include maximum likelihood estimation and statistical tests for the different configurations, (M)ANOVA, parametric Discriminant Analysis, and Model-based Clustering. Moreover, outlier detection and estimation based on robust techniques are provided; discriminant parametric methods based on robust estimates are implemented accordingly.

MAINT.Data, currently in its 2.6.1 version, offers an integrated solution for the management and parametric analysis of interval-valued data, from aggregation to modelling, analysis and visualisation, extending the R "programming with data" paradigm to new and complex data types.

Acknowledgements

This work was financed by the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within projects UIDB/00731/2020, UIDB/50014/2020, UIDB/00315/2020 and PTDC/EEI-TEL/32454/2017.

Bibliography

M. Aitkin and G. T. Wilson. Mixture models, outliers, and the EM algorithm. *Technometrics*, 22(3):325–331, 1980. URL `https://doi.org/10.1080/00401706.1980.10486163`. [p358]

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. URL `https://doi.org/10.1109/TAC.1974.1100705`. [p339]

A. Azzalini. A class of distributions which includes the Normal ones. *Scandinavian Journal of Statistics*, 12:171–178, 1985. [p339]

A. Azzalini. The Skew-Normal distribution and related multivariate families. *Scandinavian Journal of Statistics*, 32:159–188, 2005. URL `https://doi.org/10.1111/j.1467-9469.2005.00426.x`. [p339, 340]

A. Azzalini. *The R package* `sn`*: The Skew-Normal and Related Distributions such as the Skew-t (version 2.0-0).* Università di Padova, Italia, 2021. URL `http://azzalini.stat.unipd.it/SN`. [p347]

A. Azzalini and A. Capitanio. Statistical applications of the multivariate Skew-Normal distribution. *Journal of the Royal Statistical Society Series B-Methodological*, 61(3):579–602, 1999. URL `https://doi.org/10.1111/1467-9868.00194`. [p340, 342]

A. Azzalini and A. Dalla Valle. The multivariate Skew-Normal distribution. *Biometrika*, 83 (4):715–726, 1996. URL `https://doi.org/10.1093/biomet/83.4.715`. [p336, 339]

J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, pages 803–821, 1993. URL `https://doi.org/10.2307/2532201`. [p342]

B. Brahim and S. Makosso-Kallyth. *GPCSIV: Generalized Principal Component of Symbolic Interval variables*, 2013. URL `https://CRAN.R-project.org/package=GPCSIV`. R package version 0.1.0. [p337]

P. Brito. Symbolic Data Analysis: another look at the interaction of Data Mining and Statistics. *WIREs Data Mining and Knowledge Discovery*, 4(4):281–295, 2014. URL `https://doi.org/10.1002/widm.1133`. [p336, 337]

P. Brito and A. P. Duarte Silva. Modelling interval data with Normal and Skew-Normal distributions. *Journal of Applied Statistics*, 39(1):3–20, 2012. URL `https://doi.org/10.1080/02664763.2011.575125`. [p336, 337, 338, 339, 340, 341, 343]

P. Brito, A. P. Duarte Silva, and J. Dias. Probabilistic clustering of interval data. *Intelligent Data Analysis*, 19(2):293–313, 2015. URL `https://doi.org/10.3233/IDA-150718`. [p337, 343]

A. Cerioli. Multivariate outlier detection with high-breakdown estimators. *Journal of the American Statistical Association*, 105(489):147–156, 2010. URL `https://doi.org/10.1198/jasa.2009.tm09147`. [p340, 341]

F. De Carvalho and Y. Lechevallier. Partitional clustering algorithms for symbolic interval data based on single adaptive distances. *Pattern Recognition*, 42(7):1223–1236, 2009. URL `https://doi.org/10.1016/j.patcog.2008.11.016`. [p337]

F. De Carvalho, P. Brito, and H.-H. Bock. Dynamic clustering for interval data based on L_2 distance. *Computational Statistics*, 21(2):231–250, 2006. URL `https://doi.org/10.1007/s00180-006-0261-z`. [p337]

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B-Methodological*, 39(1):1–38, 1977. URL https://doi.org/10.1111/j.2517-6161.1977.tb01600.x. [p343]

S. Dias and P. Brito. Off the beaten track: A new linear model for interval data. *European Journal of Operational Research*, 258(3):1118–1130, 2017. URL https://doi.org/10.1016/j.ejor.2016.09.006. [p337]

E. Diday and M. Noirhomme-Fraiture. *Symbolic Data Analysis and the SODAS Software*. John Wiley & Sons, Chichester, 2008. [p336]

A. Douzal-Chouakria, L. Billard, and E. Diday. Principal component analysis for interval-valued observations. *Statistical Analysis and Data Mining*, 4(2):229–246, 2011. URL https://doi.org/10.1002/sam.10118. [p337]

A. P. Duarte Silva and P. Brito. Discriminant analysis of interval data: An assessment of parametric and distance-based approaches. *Journal of Classification*, 32(3):516–541, 2015. URL https://doi.org/0.1007/s00357-015-9189-8. [p337, 342]

A. P. Duarte Silva and P. Brito. *MAINT.Data: Model and Analyse Interval Data*, 2021. URL https://CRAN.R-project.org/package=MAINT.Data. R package version 2.6.1. [p337]

A. P. Duarte Silva, P. Filzmoser, and P. Brito. Outlier detection in interval data. *Advances in Data Analysis and Classification*, 12(3):785–822, 2018. URL https://doi.org/10.1007/s11634-017-0305-y. [p337, 340, 341]

A. Dudek, M. Pelka, J. Wilk, and M. Walesiak. *symbolicDA: Analysis of Symbolic Data*, 2019. URL https://CRAN.R-project.org/package=symbolicDA. R package version 0.6-2. [p337]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p347]

C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002. URL https://doi.org/10.1198/016214502760047131. [p342]

R. François, D. Eddelbuettel, and D. Bates. *RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library, R package (version 0.10.6.0.0)*, 2021. URL https://cran.r-project.org/web/packages/RcppArmadillo/index.html. [p347]

D. M. Gay. Usage summary for selected optimization routine. Technical Report 153, AT&T Bell Laboratories, Murray Hill, 1990. [p347]

A. Hadi and A. Luceño. Maximum trimmed likelihood estimators: a unified approach, examples, and algorithms. *Computational Statistics & Data Analysis*, 25(3):251–272, 1997. URL https://doi.org/10.1016/S0167-9473(97)00011-X. [p340]

J. Hardin and D. Rocke. The distribution of robust distances. *Journal of Computational and Graphical Statistics*, 14:910–927, 2005. URL https://doi.org/10.1198/106186005X77685. [p340, 341]

S. Hubeaux and K. Rufibach. *SurvRegCensCov: Weibull Regression for a Right-Censored Endpoint with Interval-Censored Covariate*, 2015. URL https://CRAN.R-project.org/package=SurvRegCensCov. R package version 1.4. [p337]

M. Hubert, P. Rousseeuw, and S. Van Aelst. High-breakdown robust multivariate methods. *Statistical Science*, 23(1):92–119, 2008. URL https://doi.org/10.1214/088342307000000087. [p340]

C. J. Huberty and S. Olejnik. *Applied MANOVA and Discriminant Analysis, 2nd Edition*. John Wiley & Sons, Chichester, 2006. [p341]

J. Le-Rademacher and L. Billard. Likelihood functions and some maximum likelihood estimators for symbolic data. *Journal of Statistical Planning and Inference*, 141(4):1593–1602, 2011. URL https://doi.org/10.1016/j.jspi.2010.11.016. [p337]

J. Le-Rademacher and L. Billard. Symbolic covariance principal component analysis and visualization for interval-valued data. *Journal of Computational and Graphical Statistics*, 21 (2):413–432, 2012. URL https://doi.org/10.1080/10618600.2012.679895. [p337]

E. Lima Neto and F. De Carvalho. Centre and range method for fitting a linear regression model to symbolic interval data. *Computational Statistics & Data Analysis*, 52(3):1500–1515, 2008. URL https://doi.org/10.1016/j.csda.2007.04.014. [p337]

E. Lima Neto and F. De Carvalho. Constrained linear regression models for symbolic interval-valued variables. *Computational Statistics & Data Analysis*, 54(2):333–347, 2010. URL https://doi.org/10.1016/j.csda.2009.08.010. [p337]

E. Lima Neto and F. De Carvalho. Bivariate symbolic regression models for interval-valued variables. *Journal of Statistical Computation and Simulation*, 81(11):1727–1744, 2011. URL https://doi.org/10.1080/00949655.2010.500470. [p337]

E. Lima Neto, C. De Souza Filho, and P. Marinho. *iRegression: Regression Methods for Interval-Valued Variables*, 2016. URL https://CRAN.R-project.org/package=iRegression. R package version 1.2.1. [p337]

G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000. [p342]

G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, 1992. [p342]

J. Messner, A. Zeileis, and R. Stauffer. *crch: Censored Regression with Conditional Heteroscedasticity*, 2019. URL https://CRAN.R-project.org/package=crch. R package version 1.0-4. [p337]

G. Pison, S. Van Aelst, and G. Willems. Small sample corrections for LTS and MCD. *Metrika*, 55(1-2):111–123, 2002. URL https://doi.org/10.1007/s001840200191. [p340]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p337]

A. B. Ramos-Guajardo and P. Grzegorzewski. Distance-based linear discriminant analysis for interval-valued data. *Information Sciences*, 372:591–607, 2016. URL https://doi.org/10.1016/j.ins.2016.08.068. [p337]

O. Rodriguez. *RSDA: R to Symbolic Data Analysis*, 2021. URL https://CRAN.R-project.org/package=RSDA. R package version 3.0.9. [p337]

P. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984. URL https://doi.org/10.2307/2288718. [p340]

P. Rousseeuw. Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, and W. Wertz, editors, *Mathematical Statistics and Applications*, pages 283–297. Reidel Publishing, 1985. [p340]

P. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999. URL https://doi.org/10.1080/00401706.1999.10485670. [p340, 348]

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978. URL https://doi.org/10.1214/aos/1176344136. [p339]

D. W. Scott. *Multivariate Density Estimation. Theory, Practice, and Visualization*. John Wiley & Sons, 2015. [p359]

L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using gaussian finite mixture models. *The R Journal*, 8(1):205–233, 2016. URL https://doi.org/10.32614/RJ-2016-021. [p348, 355]

G. A. Seber. *Multivariate Observations*, volume 252. John Wiley & Sons, 2009. [p348]

V. Todorov and P. Filzmoser. An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32(3):1–47, 2009. URL https://doi.org/10.18637/jss.v032.i03. [p348]

R. B. Valle and A. Azzalini. The centred parametrization for the multivariate Skew-Normal distribution. *Journal of Mutivariate Analysis*, 99(4):1362–1382, 2008. URL https://doi.org/10.1016/j.jmva.2008.01.020. [p340, 347]

A. Pedro Duarte Silva
Católica Porto Business School & CEGE
Universidade Católica Portuguesa
Rua Diogo Botelho, 1327
4169-005 Porto, Portugal
E-mail: psilva@ucp.pt

Paula Brito
Faculdade de Economia, Universidade do Porto
& LIAAD-INESC TEC
Rua Dr. Roberto Frias
4200-464 Porto, Portugal
E-mail: mpbrito@fep.up.pt

Peter Filzmoser
Institute of Statistics and Mathematical Methods in Economics
TU Wien
Wiedner Hauptstraße 8-10
1040 Vienna, Austria
E-mail: Peter.Filzmoser@tuwien.ac.at

José G. Dias
Business Research Unit (BRU-IUL)
Instituto Universitário de Lisboa (ISCTE-IUL)
Av. das Forças Armadas
1648-026 Lisboa, Portugal
E-mail: jose.dias@iscte-iul.pt

Robust and Efficient Optimization Using a Marquardt-Levenberg Algorithm with R Package marqLevAlg

by Viviane Philipps, Boris P. Hejblum, Mélanie Prague, Daniel Commenges and Cécile Proust-Lima

Abstract Implementations in R of classical general-purpose algorithms for local optimization generally have two major limitations which cause difficulties in applications to complex problems: too loose convergence criteria and too long calculation time. By relying on a Marquardt-Levenberg algorithm (MLA), a Newton-like method particularly robust for solving local optimization problems, we provide with marqLevAlg package an efficient and general-purpose local optimizer which (i) prevents convergence to saddle points by using a stringent convergence criterion based on the relative distance to minimum/maximum in addition to the stability of the parameters and of the objective function; and (ii) reduces the computation time in complex settings by allowing parallel calculations at each iteration. We demonstrate through a variety of cases from the literature that our implementation reliably and consistently reaches the optimum (even when other optimizers fail) and also largely reduces computational time in complex settings through the example of maximum likelihood estimation of different sophisticated statistical models.

Introduction

Optimization is an essential task in many computational problems. In statistical modeling, for instance, in the absence of analytical solutions, maximum likelihood estimators are often retrieved using iterative optimization algorithms, which locally solve the problem from given starting values.

Steepest descent algorithms are among the most famous general local optimization algorithms. They generally consist in updating parameters according to the steepest gradient (gradient descent) possibly scaled by the Hessian in the Newton (Newton-Raphson) algorithm or an approximation of the Hessian based on the gradients in the quasi-Newton algorithms (e.g., Broyden-Fletcher-Goldfarb-Shanno - BFGS). Newton-like algorithms have been shown to provide good convergence properties (Joe and Nash, 2003) and were demonstrated in particular to behave better than Expectation-Maximization (EM) algorithms in several contexts of Maximum Likelihood Estimation, such as the random-effect models (Lindstrom and Bates, 1988) or the latent class models (Proust and Jacqmin-Gadda, 2005). Among Newton methods, the Marquardt-Levenberg algorithm, initially proposed by Levenberg (Levenberg, 1944), then Marquardt (Marquardt, 1963), combines BFGS and gradient descent methods to provide a more robust optimization algorithm. Like other Newton methods, the Marquardt-Levenberg algorithm is designed to find a local optimum of the objective function from given initial values. When dealing with multimodal objective functions, it can thus converge to local optimum and needs to be combined with a grid search to retrieve the global optimum.

The R software includes multiple solutions for local and global optimization tasks (see CRAN task View on *Optimization* (Theussl et al., 2014)). In particular, the `optim` function in **base** R offers different algorithms for general-purpose optimization, and so does **optimx**, a more recent package extending `optim` (Nash and Varadhan, 2011). Numerous additional packages are available for different contexts, from nonlinear least square problems (including some exploiting Marquardt-Levenberg idea like **minpack.lm** (Elzhov et al., 2016) and **nlmrt** (Nash, 2016)) to stochastic optimization and algorithms based on the simplex approach. However, R software could benefit from a general-purpose R implementation of the Marquardt-Levenberg algorithm.

We present here an R implementation of the Marquardt-Levenberg algorithm in the package **marqLevAlg**, which relies on a stringent convergence criterion based on the first and second derivatives to avoid loosely convergence (Prague et al., 2012) and includes (from version 2.0.1) parallel computations within each iteration to speed up convergence. This implementation is particularly dedicated to complex settings, that is, when a large number of parameters are optimized, and/or the computation of the objective function is time-consuming. The parallel computations speed up the procedure, and the stringent convergence criterion prevents false convergences on the flat regions of the objective function obtained with convergence criteria based on the function stability.

Section 2 and 3 describe the algorithm and the implementation, respectively. Then Section 4 provides an example of a call with the estimation of a linear mixed model. A benchmark of the package is reported in Section 5 with the performances of parallel implementation. Performances of Marquardt-Levenberg algorithm implementation are also challenged in Section 6 using a variety of simple and complex examples from the literature and compared with other optimizers. Finally, Section 7 concludes.

Methodology

The Marquardt-Levenberg algorithm

The Marquardt-Levenberg algorithm (MLA) can be used for any problem where a function $\mathcal{F}(\theta)$ has to be minimized (or equivalently, function $\mathcal{L}(\theta)$= - $\mathcal{F}(\theta)$ has to be maximized) according to a set of m unconstrained parameters θ as long as the second derivatives of $\mathcal{F}(\theta)$ exist. In statistical applications, for instance, the objective function is the deviance to be minimized or the log-likelihood to be maximized.

Our improved MLA iteratively updates the vector $\theta^{(k)}$ from a starting point $\theta^{(0)}$ until convergence using the following formula at iteration $k+1$:

$$\theta^{(k+1)} = \theta^{(k)} - \delta_k \left(\tilde{H}(\mathcal{F}(\theta^{(k)}))\right)^{-1} \nabla(\mathcal{F}(\theta^{(k)})),$$

where $\theta^{(k)}$ is the set of parameters at iteration k, $\nabla(\mathcal{F}(\theta^{(k)}))$ is the gradient of the objective function at iteration k, and $\tilde{H}(\mathcal{F}(\theta^{(k)}))$ is the Hessian matrix $H(\mathcal{F}(\theta^{(k)}))$ where the diagonal terms are replaced by $\tilde{H}(\mathcal{F}(\theta^{(k)}))_{ii} = H(\mathcal{F}(\theta^{(k)}))_{ii} + \lambda_k[(1-\eta_k)|H(\mathcal{F}(\theta^{(k)}))_{ii}| + \eta_k \text{tr}(H(\mathcal{F}(\theta^{(k)})))]$. In the original MLA, the Hessian matrix is inflated by a scaled identity matrix. Following Fletcher (1971), we consider refined inflation based on the curvature. The diagonal inflation of our improved MLA makes it an intermediate between the steepest descent method and the Newton method. The parameters δ_k, λ_k, and η_k are scalars specifically determined at each iteration k. Parameter δ_k is fixed to 1 unless the objective function is not reduced, in which case a line search determines the locally optimal step length. Parameters λ_k and η_k are internally modified in order to ensure that (i) $\tilde{H}(\mathcal{F}(\theta^{(k)}))$ be definite-positive at each iteration k, and (ii) $\tilde{H}(\mathcal{F}(\theta^{(k)}))$ approaches $H(\mathcal{F}(\theta^{(k)}))$ when $\theta^{(k)}$ approaches $\hat{\theta}$.

When the problem encounters a unique solution, the minimum is reached whatever the chosen initial values.

Stringent convergence criteria

As in any iterative algorithm, the convergence of MLA is achieved when convergence criteria are fulfilled. In **marqLevAlg** package, convergence is defined according to three criteria:

- parameters stability: $\sum_{j=1}^{m} \left(\theta_j^{(k+1)} - \theta_j^{(k)}\right)^2 < \epsilon_a$;
- objective function stability: $|\mathcal{F}^{(k+1)} - \mathcal{F}^{(k)}| < \epsilon_b$;
- relative distance to minimum/maximum (RDM): $\frac{\nabla(\mathcal{F}(\theta^{(k)}))\left(H(\mathcal{F}(\theta^{(k)}))\right)^{-1}\nabla(\mathcal{F}(\theta^{(k)}))}{m} < \epsilon_d$.

The original Marquardt-Levenberg algorithm (Marquardt, 1963) and its implementations (Elzhov et al., 2016; Nash, 2016) consider the two first criteria (as well as a third one based on the angle between the objective function and its gradient). Yet, these criteria, which are also used in many other iterative algorithms, do not ensure convergence towards an actual optimum. They only ensure the convergence towards a saddle point. We thus chose to complement the parameter and objective function stability by the relative distance to minimum/maximum. As it requires the Hessian matrix to be invertible, it prevents any convergence to a saddle point and is thus essential to ensure that an optimum is truly reached. When the Hessian is not invertible, RDM is set to 1+ϵ_d, and convergence criteria cannot be fulfilled.

Although it constitutes a relevant convergence criterion in any optimization context, RDM was initially designed for log-likelihood maximization problems, that is, cases where $\mathcal{F}(\theta)$= - $\mathcal{L}(\theta)$ with $\mathcal{L}$ the log-likelihood. In that context, RDM can be interpreted as the ratio between the numerical error and the statistical error (Commenges et al., 2006, Prague et al. (2013)).

The three thresholds ϵ_a, ϵ_b, and ϵ_d can be adjusted, but values around 0.0001 are usually sufficient to guarantee a correct convergence. In some complex log-likelihood maximization problems, for instance, Prague et al. (2013) showed that the RDM convergence properties remain acceptable, providing ϵ_d is below 0.1 (although the lower, the better).

Derivatives calculation

MLA update relies on first ($\nabla(\mathcal{F}(\theta^{(k)}))$) and second ($H(\mathcal{F}(\theta^{(k)}))$) derivatives of the objective function $\mathcal{F}(\theta^{(k)})$ at each iteration k. The gradient and the Hessian may sometimes be calculated analytically, but numerical approximation can become necessary in a general framework. In **marqLevAlg** package, in the absence of analytical gradient computation, the first derivatives are computed by central finite differences. In the absence of analytical Hessian, the second derivatives are computed using forward finite differences. The step of finite difference for each derivative depends on the value of the involved parameter. It is set to $\max(10^{-7}, 10^{-4}|\theta_j|)$ for parameter j.

When both the gradient and the Hessian are to be numerically computed, numerous evaluations of $\mathcal{F}$ are required at each iteration:

- $2 \times m$ evaluations of $\mathcal{F}$ for the numerical approximation of the gradient function;
- $\dfrac{m \times (m+1)}{2}$ evaluations of $\mathcal{F}$ for the numerical approximation of the Hessian matrix.

The number of derivatives thus grows quadratically with the number m of parameters, and calculations are per se independent as done for different vectors of parameters θ.

When the gradient is analytically calculated, only the second derivatives have to be approximated, requiring $2 \times m$ independent calls to the gradient function. In that case, the complexity thus linearly increases with m.

In both cases, and especially when each calculation of derivative is long and/or m is large, parallel computations of independent $\mathcal{F}$ evaluations become particularly relevant to speed up the estimation process.

Special case of a log-likelihood maximization

When the optimization problem is the maximization of the log-likelihood $\mathcal{L}(\theta)$ of a statistical model according to parameters θ, the Hessian matrix of the $\mathcal{F}(\theta) = -\mathcal{L}(\theta)$ calculated at the optimum $\hat{\theta}$, $\mathcal{H}(\mathcal{F}(\hat{\theta})) = -\left.\dfrac{\partial^2 \mathcal{L}(\theta)}{\partial \theta^2}\right|_{\theta=\hat{\theta}}$, provides an estimator of the Fisher Information matrix. The inverse of $\mathcal{H}(\mathcal{F}(\hat{\theta}))$ computed in the package thus provides an estimator of the variance-covariance matrix of the optimized vector of parameters $\hat{\theta}$.

Implementation

marqLevAlg function

The call of the `marqLevAlg` function, or its shorcut `mla`, is the following :

```
marqLevAlg(b, m = FALSE, fn, gr = NULL, hess = NULL, maxiter = 500,
  epsa = 0.0001, epsb = 0.0001, epsd = 0.0001, digits = 8,
  print.info = FALSE, blinding = TRUE, multipleTry = 25, nproc = 1,
  clustertype = NULL, file = "", .packages = NULL, minimize = TRUE, ...)
```

Argument `b` is the set of initial parameters; alternatively, its length `m` can be entered. `fn` is the function to optimize; it should take the parameter vector as the first argument, and additional arguments are passed in .... Optional `gr` and `hess` refer to the functions implementing the analytical calculations of the gradient and the Hessian matrix, respectively. `maxiter` is the maximum number of iterations. Arguments `epsa`, `epsb`, and `epsd` are the thresholds for the three convergence criteria defined in Section 2.2.2. `print.info` specifies if details on each iteration should be printed; such information can be reported in a file if argument `file` is specified, and `digits` indicates the number of decimals in the eventually reported information during optimization. `blinding` is an option allowing the algorithm to go on even when the `fn` function returns NA, which is then replaced by the arbitrary value of 500,000 (for minimization) and -500,000 (for maximization). Similarly, if an infinite value is found for the chosen initial values, the `multipleTry` option will internally reshape `b` (up to `multipleTry` times) until a finite value is got, and the algorithm can be correctly initialized. The parallel framework is first stated by the `nproc` argument, which gives the number of cores, and by the `clustertype` argument (see the next section). In the case where the `fn` function depends on R packages, these should be given as a character vector in the `.packages` argument. Finally, the `minimize` argument offers the possibility to minimize or maximize the objective function `fn`; a maximization problem is implemented as the minimization of the opposite function (`-fn`).

Implementation of parallel computations

In the absence of analytical gradient calculation, derivatives are computed in the `deriva` subfunction with two loops, one for the first derivatives and one for the second derivatives. Both loops are parallelized. The parallelized loops are at most over $m * (m+1)/2$ elements for m parameters to estimate, which suggests that the performance could theoretically be improved with up to $m * (m+1)/2$ cores.

When the gradient is calculated analytically, the `deriva` subfunction is replaced by the `deriva_grad` subfunction. It is parallelized in the same way, but with the parallelization being executed over m elements, the performance should be bounded at m cores.

In all cases, parallelization is achieved using the **doParallel** and **foreach** packages. The snow and multicore options of the `doParallel` backend are kept, making the parallel option of **marqLevAlg** package available on all systems. The user specifies the type of parallel environment among FORK, SOCK, or MPI in argument `clustertype` and the number of cores in `nproc`. For instance, `clustertype = "FORK",nproc = 6` will use FORK technology and 6 cores.

Example

We illustrate how to use `marqLevAlg` function with the maximum likelihood estimation in a linear mixed model (Laird and Ware, 1982). Function `loglikLMM` available in the package implements the log-likelihood of a linear mixed model for a dependent outcome vector ordered by subject (argument Y) explained according to a matrix of covariates (argument X) entered in the same order as Y with a Gaussian individual-specific random intercept and Gaussian independent errors:

```
loglikLMM(b, Y, X, ni)
```

Argument b specifies the vector of parameters with first the regression parameters (length given by the number of columns in X) and then the standard deviations of the random intercept and the independent error. Finally, argument ni specifies the number of repeated measures for each subject.

We consider the dataset `dataEx` (available in the package) in which variable Y is repeatedly observed at time t for 500 subjects along with a binary variable $X1$ and a continuous variable $X3$. For the illustration, we specify a linear trajectory over time adjusted for $X1$, $X3$, and the interaction between $X1$ and time t. The vector of parameters to estimate corresponds to the intercept, 4 regression parameters, and the 2 standard deviations.

We first define the quantities to include as an argument in `loglikLMM` function:

```
> Y <- dataEx$Y
> X <- as.matrix(cbind(1, dataEx[, c("t", "X1", "X3")],
+                      dataEx$t * dataEx$X1))
> ni <- as.numeric(table(dataEx$i))
```

The vector of initial parameters to specify in `marqLevAlg` call is created with the trivial values of 0 for the fixed effects and 1 for the variance components.

```
> binit <- c(0, 0, 0, 0, 0, 1, 1)
```

The maximum likelihood estimation of the linear mixed model in sequential mode is then run using a simple call to `marqLevAlg` function for a maximization (with argument `minimize = FALSE`):

```
> estim <- marqLevAlg(b = binit, fn = loglikLMM, minimize = FALSE,
+                     X = X, Y = Y, ni = ni)
> estim

                 Robust marqLevAlg algorithm

marqLevAlg(b = binit, fn = loglikLMM, minimize = FALSE, X = X,
    Y = Y, ni = ni)

Iteration process:
      Number of parameters: 7
      Number of iterations: 18
      Optimized objective function: -6836.754
      Convergence criteria satisfied
Convergence criteria: parameters stability= 3.2e-07
                    : objective function stability= 4.35e-06
                    : Matrix inversion for RDM successful
                    : relative distance to maximum(RDM)= 0

Final parameter values:
 50.115  0.106  2.437  2.949 -0.376 -5.618  3.015
```

The printed output estim shows that the algorithm converged in 18 iterations with convergence criteria of 3.2e-07, 4.35e-06, and 0 for parameters stability, objective function stability, and RDM, respectively. The output also displays the list of coefficient values at the optimum. All this information can also be recovered in the estim object, where item b contains the estimated coefficients.

As mentioned in Section 2.2.4, in log-likelihood maximization problems, the inverse of the Hessian given by the program provides an estimate of the variance-covariance matrix of the coefficients at the optimum. The upper triangular matrix of the inverse Hessian is thus systematically computed in object v. When appropriate, the summary function can output this information with option loglik = TRUE. With this option, the summary also includes the square root of these variances (i.e., the standards errors), the corresponding Wald statistic, the associated *p*-value, and the 95% confidence interval boundaries for each parameter:

```
> summary(estim, loglik = TRUE)

                   Robust marqLevAlg algorithm

marqLevAlg(b = binit, fn = loglikLMM, minimize = FALSE, X = X,
    Y = Y, ni = ni)

Iteration process:
      Number of parameters: 7
      Number of iterations: 18
      Optimized objective function: -6836.754
      Convergence criteria satisfied

Convergence criteria: parameters stability= 3.2e-07
                    : objective function stability= 4.35e-06
                    : Matrix inversion for RDM successful
                    : relative distance to maximum(RDM)= 0

Final parameter values:
   coef SE.coef        Wald P.value   binf   bsup
 50.115   0.426 13839.36027   0e+00 49.280 50.950
  0.106   0.026    16.02319   6e-05  0.054  0.157
  2.437   0.550    19.64792   1e-05  1.360  3.515
  2.949   0.032  8416.33202   0e+00  2.886  3.012
 -0.376   0.037   104.82702   0e+00 -0.449 -0.304
 -5.618   0.189   883.19775   0e+00 -5.989 -5.248
  3.015   0.049  3860.64370   0e+00  2.919  3.110
```

The exact same model can also be estimated in a parallel mode using FORK implementation of parallelism (here with two cores):

```
> estim2 <- marqLevAlg(b = binit, fn = loglikLMM, minimize = FALSE,
+                      nproc = 2, clustertype = "FORK",
+                      X = X, Y = Y, ni = ni)
```

It can also be estimated by using analytical gradients (provided in gradient function gradLMM with the same arguments as loglikLMM):

```
> estim3 <- marqLevAlg(b = binit, fn = loglikLMM, gr = gradLMM,
+                      minimize = FALSE, X = X, Y = Y, ni = ni)
```

In all three situations, the program converges to the same maximum as shown in Table 1 for the estimation process and Table 2 for the parameter estimates. The iteration process is identical when using either the sequential or the parallel code (number of iterations, final convergence criteria, etc.). It necessarily differs slightly when using the analytical gradient, as the computation steps are not identical (e.g., here it converges in 15 iterations rather than 18), but all the final results are identical.

	Object estim	Object estim2	Object estim3
Number of cores	1	2	1
Analytical gradient	no	no	yes
Objective Function	-6836.754	-6836.754	-6836.754
Number of iterations	18	18	15
Parameter Stability	3.174428e-07	3.174428e-07	6.633702e-09
Likelihood stability	4.352822e-06	4.352822e-06	9.159612e-08
RDM	1.651774e-12	1.651774e-12	2.935418e-17

Table 1: Summary of the estimation process of a linear mixed model using 'marqLevAlg' function run either in sequential mode with numerical gradient calculation (object estim), parallel mode with numerical gradient calculation (object estim2), or sequential mode with analytical gradient calculation (object estim3).

	Object estim		Object estim2		Object estim3	
	Coef	SE	Coef	SE	Coef	SE
Parameter 1	50.1153	0.4260	50.1153	0.4260	50.1153	0.4260
Parameter 2	0.1055	0.0264	0.1055	0.0264	0.1055	0.0264
Parameter 3	2.4372	0.5498	2.4372	0.5498	2.4372	0.5498
Parameter 4	2.9489	0.0321	2.9489	0.0321	2.9489	0.0321
Parameter 5	-0.3764	0.0368	-0.3764	0.0368	-0.3764	0.0368
Parameter 6	-5.6183	0.1891	-5.6183	0.1891	5.6183	0.1891
Parameter 7	3.0145	0.0485	3.0145	0.0485	3.0145	0.0485

Table 2: Estimates (Coef) and standard error (SE) of the parameters of a linear mixed model fitted using 'marqLevAlg' function run either in sequential mode with numerical gradient calculation (object estim), parallel mode with numerical gradient calculation (object estim2), or sequential mode with analytical gradient calculation (object estim3).

Benchmark

We aimed at evaluating and comparing the performances of the parallelization in some time-consuming examples. We focused on three examples of sophisticated models from the mixed models area, estimated by maximum likelihood. These examples rely on packages using three different languages, thus illustrating the behavior of **marqLevAlg** package with a program exclusively written in R (**JM**, Rizopoulos (2010)) and programs, including Rcpp (**CInLPN**, Taddé et al. (2019)) and Fortran90 (**lcmm**, Proust-Lima et al. (2017)) languages, widely used in complex situations.

We first describe the generated dataset on which the benchmark has been realized. We then introduce each statistical model and associated program. Finally, we detail the results obtained with the three programs. Each time, the model has been estimated sequentially and with a varying number of cores in order to provide the program speed-up. We used a Linux cluster with 32 cores machines and 100 replicates to assess the variability. Codes and datasets used in this section are available at `https://github.com/VivianePhilipps/marqLevAlgPaper.`

Simulated dataset

We generated a dataset of 20,000 subjects having repeated measurements of a marker `Ycens` (measured at times `t`) up to a right-censored time of event `tsurv` with the indicator that the event occurred `event`. The data were generated according to a 4 latent class joint model (Proust-Lima et al., 2014). This model assumes that the population is divided into 4 latent classes, each class having a specific trajectory of the marker defined according to a linear mixed model with specific parameters, and specific risk of event defined according to a parametric proportional hazard model with specific parameters too. The class-specific linear mixed model included a basis of natural cubic splines with 3 equidistant knots taken at times 5, 10 and 15, associated with fixed and correlated random effects. The proportional hazard model included a class-specific Weibull risk adjusted on 3 covariates: one binary (Bernoulli with 50% probability) and two continous variables (standard Gaussian and Gaussian with mean 45 and standard deviation 8). The proportion of individuals in each class is about 22%, 17%, 34%, and 27% in the sample.

Below are given the five first rows of the three first subjects:

```
   i class X1        X2       X3 t    Ycens     tsurv event
1  1     2  0 0.6472205 43.42920 0 61.10632 20.000000     0
2  1     2  0 0.6472205 43.42920 1 60.76988 20.000000     0
3  1     2  0 0.6472205 43.42920 2 58.72617 20.000000     0
4  1     2  0 0.6472205 43.42920 3 56.76015 20.000000     0
5  1     2  0 0.6472205 43.42920 4 54.04558 20.000000     0
22 2     1  0 0.3954846 43.46060 0 37.95302  3.763148     1
23 2     1  0 0.3954846 43.46060 1 34.48660  3.763148     1
24 2     1  0 0.3954846 43.46060 2 31.39679  3.763148     1
25 2     1  0 0.3954846 43.46060 3 27.81427  3.763148     1
26 2     1  0 0.3954846 43.46060 4       NA  3.763148     1
43 3     3  0 1.0660837 42.08057 0 51.60877 15.396958     1
44 3     3  0 1.0660837 42.08057 1 53.80671 15.396958     1
45 3     3  0 1.0660837 42.08057 2 51.11840 15.396958     1
46 3     3  0 1.0660837 42.08057 3 50.64331 15.396958     1
47 3     3  0 1.0660837 42.08057 4 50.87873 15.396958     1
```

Statistical models

Joint shared random effect model for a longitudinal marker and a time to event: package JM

The maximum likelihood estimation of joint shared random effect models has been made available in R with the **JM** package (Rizopoulos, 2010). The implemented optimization functions are `optim` and `nlminb`. We added the `marqLevALg` function for the purpose of this example. We considered a subsample of the simulated dataset, consisting of 5,000 randomly selected subjects.

The joint shared random effect model is divided into two submodels jointly estimated:

- a linear mixed submodel for the repeated marker Y measured at different times t_{ij} ($j = 1, ..., n_i$):

$$\begin{aligned} Y_i(t_{ij}) &= \tilde{Y}_i(t_{ij}) + \varepsilon_{ij} \\ &= X_i(t_{ij})\beta + Z_i(t_{ij})u_i + \varepsilon_{ij}, \end{aligned}$$

where, in our example, $X_i(t)$ contained the intercept, the class indicator, the 3 simulated covariates, a basis of natural cubic splines on time t (with 2 internal knots at times 5 and 15), and the interactions between the splines and the time-invariant covariates, resulting in 20 fixed effects. $Z_i(t)$ contained the intercept and the same basis of natural cubic splines on time t, and was associated with u_i, the 4-vector of correlated Gaussian random effects. ε_{ij} was the independent Gaussian error.

- a survival submodel for the right censored time-to-event:

$$\alpha_i(t) = \alpha_0(t)\exp(X_{si}\gamma + \eta\tilde{Y}_i(t)),$$

where, in our example, the vector X_{si}, containing the 3 simulated covariates, was associated with the vector of parameters γ; the current underlying level of the marker $\tilde{Y}_i(t)$ was associated with parameter η and the baseline hazard $\alpha_0(t)$ was defined using a basis of B-splines with 1 interior knot.

The length of the total vector of parameters θ to estimate was 40 (20 fixed effects and 11 variance component parameters in the longitudinal submodel and 9 parameters in the survival submodel).

One particularity of this model is that the log-likelihood does not have a closed-form. It involves an integral over the random effects (here, of dimension 4), which is numerically computed using an adaptive Gauss-Hermite quadrature with 3 integration points for this example.

As package **JM** includes an analytical computation of the gradient, we ran two estimations: one with the analytical gradient and one with the numerical approximation to compare the speed up and execution times.

Latent class linear mixed model: package lcmm

The second example is a latent class linear mixed model, as implemented in the `hlme` function of the **lcmm** R package. The function uses a previous implementation of the Marquardt algorithm coded in Fortran90 and in sequential mode. For the purpose of this example, we extracted the log-likelihood computation programmed in Fortran90 to be used with **marqLevAlg** package.

The latent class linear mixed model consists of two submodels estimated jointly:

- a multinomial logistic regression for the latent class membership (c_i):

$$\mathbb{P}(c_i = g) = \frac{\exp(W_i \zeta_g)}{\sum_{l=1}^{G} \exp(W_i \zeta_l)} \qquad , \quad \text{with } g = 1, ..., G,$$

where $\zeta_G = 0$ for identifiability, and W_i contained an intercept and the 3 covariates.

- a linear mixed model specific to each latent class g for the repeated outcome Y measured at times t_{ij} ($j = 1, ..., n_i$):

$$Y_i(t_{ij}|c_i = g) = X_i(t_{ij})\beta_g + Z_i(t_{ij})u_{ig} + \varepsilon_{ij},$$

where, in this example, $X_i(t)$ and $Z_i(t)$ contained an intercept, time t, and quadratic time. The vector u_{ig} of correlated Gaussian random effects had a proportional variance across latent classes, and ε_{ij} were independent Gaussian errors.

The log-likelihood of this model has a closed-form, but it involves the logarithm of a sum over latent classes, which can become computationally demanding. We estimated the model on the total sample of 20,000 subjects with 1, 2, 3, and 4 latent classes, which corresponded to 10, 18, 26, and 34 parameters to estimate, respectively.

Multivariate latent process mixed model: package CInLPN

The last example is provided by the **CInLPN** package, which relies on the Rcpp language. The function fits a multivariate linear mixed model combined with a system of difference equations in order to retrieve temporal influences between several repeated markers (Taddé et al., 2019). We used the data example provided in the package where three continuous markers `L_1`, `L_2`, `L_3` were repeatedly measured over time. The model related each marker k ($k = 1, 2, 3$) measured at observation times t_{ijk} ($j = 1, ..., T$) to its underlying level $\Lambda_{ik}(t_{ijk})$ as follows:

$$\mathrm{L}_{ik}(t_{ijk}) = \eta_{0k} + \eta_{1k}\Lambda_{ik}(t_{ijk}) + \epsilon_{ijk},$$

where ϵ_{ijk} are independent Gaussian errors and (η_0, η_1) parameters to estimate. Simultaneously, the structural model defines the initial state at time 0 ($\Lambda_{ik}(0)$) and the change over time at subsequent times t with δ is a discretization step:

$$\Lambda_{ik}(0) = \beta_{0k} + u_{ik}$$
$$\frac{\Lambda_{ik}(t+\delta) - \Lambda_{ik}(t)}{\delta} = \gamma_{0k} + v_{ik} + \sum_{l=1}^{K} a_{kl}\Lambda_{il}(t),$$

where u_{ik} and v_{ik} are Gaussian random effects.

Again, the log-likelihood of this model that depends on 27 parameters has a closed-form, but it may involve complex calculations.

Results

All the models have been estimated with 1, 2, 3, 4, 6, 8, 10, 15, 20, 25, and 30 cores. To fairly compare the execution times, we ensured that changing the number of cores did not affect the final estimation point or the number of iterations needed to converge. The mean of the speed up over the 100 replicates is reported in Table 3 and plotted in Figure 1.

The joint shared random effect model (`JM`) converged in 16 iterations after 4279 seconds in sequential mode when using the analytical gradient. Running the algorithm in parallel on 2 cores made the execution 1.85 times shorter. Computational time was gradually reduced with a number of cores

	JM		hlme				CInLPN
	analytic	numeric	G=1	G=2	G=3	G=4	
Number of parameters	40	40	10	18	26	34	27
Number of iterations	16	16	30	30	30	30	13
Number of elements in foreach loop	40	860	65	189	377	629	405
Sequential time (seconds)	4279	14737	680	3703	10402	22421	272
Speed up with 2 cores	1.85	1.93	1.78	1.93	1.94	1.96	1.89
Speed up with 3 cores	2.40	2.80	2.35	2.81	2.88	2.92	2.75
Speed up with 4 cores	2.97	3.57	2.90	3.58	3.80	3.87	3.56
Speed up with 6 cores	3.66	4.90	3.49	5.01	5.44	5.66	4.95
Speed up with 8 cores	4.15	5.84	3.71	5.84	6.90	7.26	5.96
Speed up with 10 cores	4.23	6.69	3.98	6.70	8.14	8.96	6.89
Speed up with 15 cores	4.32	7.24	3.59	7.29	10.78	12.25	8.14
Speed up with 20 cores	4.28	7.61	3.11	7.71	12.00	15.23	8.36
Speed up with 25 cores	3.76	7.29	2.60	7.37	12.30	16.84	8.11
Speed up with 30 cores	3.41	6.82	2.47	6.82	13.33	17.89	7.83

Table 3: Estimation process characteristics for the 3 different programs (JM, hlme, and CInLPN). Analytic and Numeric refer to the analytical and numerical computations of the gradient in JM; G refers to the number of latent classes.

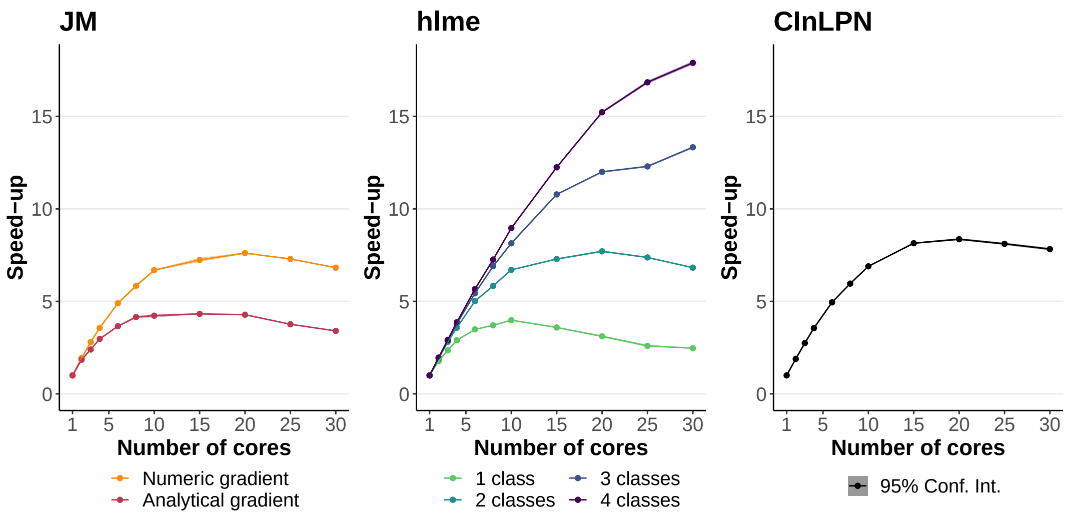


Figure 1: Speed up performances for the 3 different programs (JM, hlme, and CInLPN). Analytic and Numeric refer to the analytical and numerical computations of the gradient in JM. The number of parameters was 40 for JM; 10, 18, 26, 34 for hlme with 1, 2, 3, 4 classes, respectively; 27 for CInLPN.

between 2 and 10 to reach a maximal speed up slightly above 4. With 15, 20, 25, or 30 cores, the performances were no more improved, the speed up showing even a slight reduction, probably due to the overhead. In contrast, when the program involved numerical computations of the gradient, the parallelization reduced the computation time by a factor of almost 8 at maximum. The better speed-up performances with a numerical gradient calculation were expected since the parallel loops iterate over more elements.

The second example, the latent class mixed model estimation (`hlme`), showed an improvement of the performances as the complexity of the models increased. The simple linear mixed model (one class model), like the joint models with analytical gradient, reached a maximum speed-up of 4 with 10 cores. The two-class mixed model with 18 parameters showed a maximum speed up of 7.71 with 20 cores. Finally, the 3 and 4-class mixed models reached speed-ups of 13.33 and 17.89 with 30 cores and might still be improved with larger resources.

The running time of the third program (CInLPN) was also progressively reduced with the increasing number of cores reaching the maximal speed-up of 8.36 for 20 cores.

In these 7 examples, the speed up systematically reached almost 2 with 2 cores, and it remained interesting with 3 or 4 cores, although some variations in the speed-up performances began to be observed according to the complexity of the objective function computations. This highlights the benefit of the parallel implementation of MLA, even on personal computers. As the number of cores continued to increase, the speed-up performances varied a lot. Among our examples, the most promising situation was the one of the latent class mixed model (with a program in Fortran90) where the speed-up was up to 15 for 20 cores with the 4 class model.

Comparison with other optimization algorithms

Other Marquardt-Levenberg implementations

The Marquardt-Levenberg algorithm has been previously implemented in the context of nonlinear least squares problems in **minpack.lm** and **nlmrt**. We ran the examples provided in these two packages with `marqLevAlg` and compared the algorithms in terms of the final solution (that is, the residual sum-of-squares) and runtime. Results are shown in the supplementary material. Our implementation reached exactly the same value as the two others but performed slower in these simple examples.

We also compared the sensitivity to initial values of `marqLevAlg` with **minpack.lm** using a simple example from **minpack.lm**. We ran the two implementations of MLA on 100 simulated datasets, each one from 100 different starting points (see supplementary material). On the 10000 runs, `marqLevAlg` converged in 51.55% of the cases whereas the **minpack.lm** converged in 65.98% of the cases. However, 1660 estimations that converged according to nls.lm criteria were far from the effective optimum. This reduced the proportion of satisfying convergences with **minpack.lm** to 49.38% (so similar rate as `marqLevAlg`) but more importantly illustrated the convergence to saddle points when using classical convergence criteria. In contrast, all the convergences with **marqLevAlg** were closed to the effective solution thanks to its stringent RDM convergence criterion.

Examples from the literature

We tested our algorithm on 35 optimization problems designed by More et al. (1981) to test unconstrained optimization software and compared the `marqLevAlg` performances with those of several other optimizers, namely Nelder-Mead, BFGS, conjugate gradients (CG) implemented in the `optim` function, L-BFGS-B algorithm from `optimParallel`, and `nlminb`. Each problem consists of a function to optimize from given starting points. The results are presented in supplementary material in terms of bias between the real solution and the final value of the objective function. Our implementation of MLA converged in almost all the cases (31 out of 35), and provided almost no bias. Except for nlminb, which showed similar very good performances, the other algorithms converged at least once very far from the effective objective value. In addition, Nelder-Mead and CG algorithms converged only in approximately half of the cases.

Example of complex optimization problem: Maximum Likelihood Estimation of a Joint model for longitudinal and time-to-event data

Our implementation is particularly dedicated to complex problems involving many parameters and/or complex objective function calculation. We illustrate here its performances and compare them with other algorithms for the likelihood maximization of a joint model for longitudinal and time-to-event data, as an example of complex objective function optimization.

The **JM** package (Rizopoulos (2010)), dedicated to the maximum likelihood estimation of joint models, includes several optimization algorithms, namely the BFGS of `optim` function and an expectation-maximization (EM) technique internally implemented. It thus offers a nice framework to compare the reliability of MLA to find the maximum likelihood in a complex setting with the reliability of other optimization algorithms. We used in this comparison the `prothro` dataset described in the **JM** package and elsewhere (Skrondal and Rabe-Hesketh, 2004, Andersen et al. (1993)). It consists of a randomized trial in which 488 subjects were split into two treatment arms (prednisone *versus* placebo). Repeated measures of prothrombin ratio were collected over time as well as time to death. The longitudinal part of the joint model included a linear trajectory with time in the study, an indicator of the first measurement and their interaction with the treatment group. Correlated individual random effects on the intercept and the slope with time were also included. The survival part was a proportional hazard model adjusted for the treatment group as well as the dynamics of the longitudinal outcome either through the current value of the marker or its slope or both. The baseline risk function was approximated by B-splines with one internal knot. The total number of parameters to estimate was 17 or 18 (10 for the longitudinal submodel and 7 for the survival submodel, given that only the curent value of the marker or its slope or 8 for the survival model when both the current level and the slope were considered). The marker initially ranged from 6 to 176 (mean=79.0, sd=27.3).

To investigate the consistency of the results to different dimensions of the marker, we also considered cases where the marker was rescaled by a factor 0.1 or 10. In these cases, the log-likelihood was rescaled a posteriori to the original dimension of the marker to make the comparisons possible. The starting point was systematically set at the default initial value of the `jointModel` function, which is the estimation point obtained from the separated linear mixed model and proportional hazard model.

In addition to EM and BFGS included in `JM` package, we also compared the MLA performances with those of the parallel implementation of the L-BFGS-B algorithm provided by the **optimParallel** package. Codes and dataset used in this section are available at `https://github.com/VivianePhilipps/marqLevAlgPaper`.

MLA and L-BFGS-B ran on 3 cores. MLA converged when the three criteria defined in section 2.2.2 were satisfied with tolerance 0.0001, 0.0001, and 0.0001 for the parameters, the likelihood, and the RDM, respectively. BFGS and L-BFGS-B converged when the convergence criterion on the log-likelihood was satisfied with the square root of the tolerance of the machine ($\approx 10^{-8}$). The EM algorithm converged when stability on the parameters or on the log-likelihood was satisfied with tolerance 0.0001 and around 10^{-8} (i.e., the square root of the tolerance of the machine), respectively.

Nature of dependency	Algorithm	Scaling factor	Rescaled log-likelihood	Variation of value (%)	Variation of slope (%)	Number of iterations	Time in seconds
value	BFGS	1	-13958.55	-3.73		120	27.39
value	BFGS	0.1	-13957.91	-0.01		490	116.33
value	BFGS	10	-13961.54	-9.28		91	18.16
value	LBFGSB	1	-13958.41	-3.56		289	79.07
value	LBFGSB	0.1	-13957.69	-0.11		244	67.53
value	LBFGSB	10	error				
value	EM	1	-13957.91	-0.29		66	72.44
value	EM	0.1	-13957.72	0.14		104	106.70
value	EM	10	-13957.94	-0.59		62	67.80
value	MLA	1	**-13957.69**	-0.00		7	34.37
value	MLA	0.1	-13957.69	-0.00		6	29.48
value	MLA	10	-13957.69	-0.00		17	75.48
slope	BFGS	1	-13961.41		-1.85	251	52.76
slope	BFGS	0.1	-13961.23		-1.37	391	87.61
slope	BFGS	10	-13980.90		-13.98	444	80.16
slope	LBFGSB	1	-13960.69		-0.15	266	60.29
slope	LBFGSB	0.1	-13960.70		-0.27	206	47.87
slope	LBFGSB	10	-13962.56		-2.87	823	182.20
slope	EM	1	-13960.69		0.17	170	161.64
slope	EM	0.1	-13960.69		0.02	208	196.68
slope	EM	10	-13960.70		0.08	156	159.58
slope	MLA	1	**-13960.69**		-0.00	11	48.00
slope	MLA	0.1	-13960.69		-0.00	11	48.10
slope	MLA	10	-13960.69		0.00	14	61.61
both	BFGS	1	-13951.60	15.97	-28.17	164	37.83
both	BFGS	0.1	-13949.82	2.66	-4.63	502	132.84
both	BFGS	10	-13965.25	40.31	-95.26	52	10.48
both	LBFGSB	1	-13950.04	-1.67	7.10	800	177.61
both	LBFGSB	0.1	-13949.42	-0.01	0.38	411	93.31
both	LBFGSB	10	-13985.72	67.33	-147.30	18	7.75
both	EM	1	-13949.82	4.10	-7.22	159	186.69
both	EM	0.1	-13949.44	1.68	-3.66	156	152.89
both	EM	10	-13950.46	10.67	-16.31	142	220.07
both	MLA	1	**-13949.42**	-0.00	-0.00	10	49.91
both	MLA	0.1	-13949.42	-0.00	0.00	10	51.63
both	MLA	10	-13949.42	-0.00	0.00	24	121.69

Table 4: Comparison of the convergence obtained by MLA, BFGS, LBFGSB, and EM algorithms for the estimation of a joint model for prothrobin repeated marker (scaled by 1, 0.1, or 10) and time to death when considering a dependency on the current level of prothrobin ('value'), the current slope ('slope'), or both ('both'). All the models converged correctly according to the algorithm outputs. We report the final log-likelihood rescaled to scaling factor 1 (for comparison), the percentage of variation of the association parameters ('value' and 'slope' columns) compared to the one obtained with the overall maximum likelihood with scaling 1, the number of iterations and the running time in seconds.

Table 4 compares the convergence obtained using the four optimization methods when considering a pseudo-adaptive Gauss-Hermite quadrature with 15 points. All the algorithms converged correctly according to the programs except one with L-BFGS-B, which gave an error (non-finite value) during optimization. Although the model for a given association structure is exactly the same, some differences were observed in the final maximum log-likelihood (computed in the original scale of prothrombin ratio). The final log-likelihood obtained by MLA was always the same, whatever the outcome's scaling, showing its consistency. It was also higher than the one obtained using the three other algorithms, showing that BFGS, L-BFGS-B, and, to a lesser extent, EM did not systematically converge toward the effective maximum. The difference could go up to 20 points of log-likelihood for BFGS in the example with the current slope of the marker as the association structure. The convergence also differed according to outcome's scaling with BFGS/L-BFGS-B and slightly with EM, even though, in general, the EM algorithm seemed relatively stable in this example. The less stringent convergence of BFGS/L-BFGS-B and, to a lesser extent, of EM had also consequences on the parameters estimates as roughly illustrated in Table 4 with the percentage of variation in the association parameters of prothrombin dynamics estimated in the survival model (either the current value or the current slope) in comparison with the estimate obtained using MLA which gives the overall maximum likelihood. The better performances of MLA were not at the expense of the number of iterations since MLA converged in at most 22 iterations, whereas several hundreds of iterations could be required for EM or BFGS. Note, however, that one iteration of MLA is much more computationally demanding.

Finally, for BFGS, the problem of convergence was even more apparent when the outcome was scaled by a factor 10. Indeed, the optimal log-likelihood of the model assuming a bivariate association structure (on the current level and the current slope) was worse than the optimal log-likelihood of its nested model, which assumes an association structure only on the current level (i.e., constraining the parameter for the current slope to 0). We faced the same situation with the L-BFGS-B algorithm when comparing the log-likelihoods with a bivariate association and with an association through the current slope only.

Concluding remarks

We proposed in this paper a general-purpose optimization algorithm based on a robust Marquardt-Levenberg algorithm. The program, written in R and Fortran90, is available in **marqLevAlg** R package. It provides a very nice alternative to other optimization packages available in R software such as **optim**, **roptim** (Pan, 2020), or **optimx** (Nash and Varadhan, 2011) for addressing complex optimization problems. In particular, as shown in our examples, notably the estimation of joint models, it is more reliable than classical alternatives (in particular EM, BFGS, and L-BFGS-B). This is due to the very good convergence properties of the Marquardt-Levenberg algorithm associated with very stringent convergence criteria based on the first and second derivatives of the objective function, which avoids spurious convergence at saddle points (Commenges et al., 2006).

The Marquardt-Levenberg algorithm is known for its very computationally intensive iterations due to the computation of the first and second derivatives. However, compared to other algorithms, it converges in a very small number of iterations (usually less than 30 iterations). This may not make MLA competitive in terms of running time in simple and rapid settings. However, the parallel computations of the derivatives can largely speed up the program and make it very competitive with alternatives in terms of running time in complex settings.

We chose in our implementation to rely on the RDM criterion, which is a very stringent convergence criterion. As it is based on the inverse of the Hessian matrix, it may cause non-convergence issues when some parameters are at the border of the parameter space (for instance, 0 for a parameter constrained to be positive). In that case, we recommend fixing the parameter at the border of the parameter space and running the optimization again on the rest of the parameters. In cases where the stabilities of the log-likelihood and of the parameters are considered sufficient to ensure satisfactory convergence, the program outputs might be interpreted despite a lack of convergence according to the RDM, as is done for other algorithms that only converge according to the parameter and/or objective function stability.

As with any other optimization algorithm based on the steepest descent, MLA is a local optimizer. It does not ensure the convergence of multimodal objective functions toward the global optimum. In such a context, we recommend the use of a grid search which consists in running the algorithm from a grid of (random) initial values and retaining the best result as the final solution. We illustrate in supplementary material how this technique succeeds in finding the global minimum with the Wild function of the `optim` help page.

marqLevAlg is not the first optimizer to exploit parallel computations. Other R optimizers include a parallel mode, in particular stochastic optimization packages like **DEoptim** (Mullen et al., 2011), **GA** (Scrucca, 2017), **rgenoud** (Mebane, Jr. and Sekhon, 2011), or **hydroPSO** (Zambrano-Bigiarini and Rojas, 2020). We compared these packages, the local optimizer of `optimParallel`, and **marqLevAlg** for the estimation of the linear mixed model described in Section 2.4. For this specific problem `marqLevAlg` was the fastest, followed by `optimParallel` (results shown in supplementary files).

With its parallel implementation of derivative calculations combined with very good convergence properties of MLA, **marqLevAlg** package provides a promising solution for the estimation of complex statistical models in R. We have chosen for the moment to parallelize the derivatives, which is very useful for optimization problems involving many parameters. However, we could also easily parallelize the computation of the objective function when the latter is decomposed into independent sub-computations, as is the log-likelihood computed independently on the statistical units. This alternative is currently under development.

Funding

This work was funded by French National Research Agency [grant number ANR-18-CE36-0004-01 for project DyMES].

Acknowlegdments

The computing facilities MCIA (Mésocentre de Calcul Intensif Aquitain) at the Université de Bordeaux and the Université de Pau et des Pays de l'Adour provided advice on parallel computing technologies, as well as computer time.

Bibliography

P. K. Andersen, O. Borgan, R. D. Gill, and N. Keiding. *Statistical Models Based on Counting Processes*. Springer, Paris, 1993. [p374]

D. Commenges, H. Jacqmin-Gadda, C. Proust, and J. Guedj. A Newton-Like Algorithm for Likelihood Maximization: The Robust-Variance Scoring Algorithm. *arXiv:math/0610402*, Dec. 2006. URL http://arxiv.org/abs/math/0610402. arXiv: math/0610402. [p366, 376]

T. V. Elzhov, K. M. Mullen, A.-N. Spiess, and B. Bolker. *minpack.lm: R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds*, 2016. URL https://CRAN.R-project.org/package=minpack.lm. R package version 1.2-1. [p365, 366]

R. Fletcher. A modified Marquardt subroutine for non-linear least squares. 1971. Publisher: Theoretical Physics Division, Atomic Energy Research Establishment. [p366]

H. Joe and J. C. Nash. Numerical optimization and surface estimation with imprecise function evaluations. *Statistics and Computing*, 13(3):277–286, Aug. 2003. ISSN 1573-1375. doi: 10.1023/A:1024226918553. URL https://doi.org/10.1023/A:1024226918553. [p365]

N. M. Laird and J. H. Ware. Random-Effects Models for Longitudinal Data. *Biometrics*, 38(4):963–974, 1982. ISSN 0006-341X. doi: 10.2307/2529876. URL https://www.jstor.org/stable/2529876. Publisher: [Wiley, International Biometric Society]. [p368]

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, July 1944. ISSN 0033-569X, 1552-4485. doi: 10.1090/qam/10666. URL http://www.ams.org/qam/1944-02-02/S0033-569X-1944-10666-0/. [p365]

M. J. Lindstrom and D. M. Bates. Newton—Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data. *Journal of the American Statistical Association*, 83 (404):1014–1022, Dec. 1988. ISSN 0162-1459. doi: 10.1080/01621459.1988.10478693. URL https://doi.org/10.1080/01621459.1988.10478693. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01621459.1988.10478693. [p365]

D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963. ISSN 0368-4245. doi: 10.1137/0111030. URL https://epubs.siam.org/doi/abs/10.1137/0111030. Publisher: Society for Industrial and Applied Mathematics. [p365, 366]

W. R. Mebane, Jr. and J. S. Sekhon. Genetic optimization using derivatives: The rgenoud package for R. *Journal of Statistical Software*, 42(11):1–26, 2011. URL http://www.jstatsoft.org/v42/i11/. [p376]

J. J. More, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Software*, 7(1), 3 1981. doi: 10.1145/355934.355936. URL https://www.osti.gov/biblio/6256067. [p374]

K. Mullen, D. Ardia, D. Gil, D. Windover, and J. Cline. DEoptim: An R package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011. URL http://www.jstatsoft.org/v40/i06/. [p376]

J. C. Nash. *nlmrt: Functions for Nonlinear Least Squares Solutions*, 2016. URL https://CRAN.R-project.org/package=nlmrt. R package version 2016.3.2. [p365, 366]

J. C. Nash and R. Varadhan. Unifying optimization algorithms to aid software system users: optimx for R. *Journal of Statistical Software*, 43(9):1–14, 2011. URL http://www.jstatsoft.org/v43/i09/. [p365, 376]

Y. Pan. *roptim: General Purpose Optimization in R using C++*, 2020. URL https://CRAN.R-project.org/package=roptim. R package version 0.1.5. [p376]

M. Prague, A. Diakite, and D. Commenges. Package 'marqLevAlg' - Algorithme de Levenberg-Marquardt en R : une Alternative à 'optimx' pour des Problèmes de Minimisation. In *1ères Rencontres R*, Bordeaux, France, July 2012. URL https://hal.archives-ouvertes.fr/hal-00717566. [p365]

M. Prague, D. Commenges, J. Guedj, J. Drylewicz, and R. Thiébaut. Nimrod: A program for inference via a normal approximation of the posterior in models with random effects based on ordinary differential equations. *Computer methods and programs in biomedicine*, 111(2):447–458, 2013. [p366]

C. Proust and H. Jacqmin-Gadda. Estimation of linear mixed models with a mixture of distribution for the random effects. *Computer Methods and Programs in Biomedicine*, 78(2):165–173, May 2005. ISSN 0169-2607. doi: 10.1016/j.cmpb.2004.12.004. [p365]

C. Proust-Lima, M. Séne, J. M. Taylor, and H. Jacqmin-Gadda. Joint latent class models for longitudinal and time-to-event data: A review. *Statistical methods in medical research*, 23(1):74–90, Feb. 2014. ISSN 0962-2802. doi: 10.1177/0962280212445839. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5863083/. [p370]

C. Proust-Lima, V. Philipps, and B. Liquet. Estimation of Extended Mixed Models Using Latent Classes and Latent Processes: The R Package lcmm. *Journal of Statistical Software*, 78(1):1–56, June 2017. ISSN 1548-7660. doi: 10.18637/jss.v078.i02. URL https://www.jstatsoft.org/index.php/jss/article/view/v078i02. Number: 1. [p370]

D. Rizopoulos. JM: An R package for the joint modelling of longitudinal and time-to-event data. *Journal of Statistical Software (Online)*, 35(9):1–33, July 2010. ISSN 15487660. doi: 10.18637/jss.v035.i09. URL https://repub.eur.nl/pub/89690/. [p370, 371, 374]

L. Scrucca. On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution. *The R Journal*, 9(1):187–206, 2017. URL https://journal.r-project.org/archive/2017/RJ-2017-008. [p376]

A. Skrondal and S. Rabe-Hesketh. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. CRC Press, May 2004. ISBN 978-0-203-48943-7. [p374]

B. O. Taddé, H. Jacqmin-Gadda, J.-F. Dartigues, D. Commenges, and C. Proust-Lima. Dynamic modeling of multivariate dimensions and their temporal relationships using latent processes: Application to Alzheimer's disease. *Biometrics*, n/a(n/a), 2019. ISSN 1541-0420. doi: 10.1111/biom.13168. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.13168. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/biom.13168. [p370, 372]

S. Theussl, F. Schwendinger, and H. W. Borchers. CRAN task view: Optimization and mathematical programming, 2014. URL https://cran.r-project.org/view=Optimization. Version 2020-05-21. [p365]

M. Zambrano-Bigiarini and R. Rojas. *hydroPSO: Particle Swarm Optimisation, with Focus on Environmental Models*, 2020. URL https://CRAN.R-project.org/package=hydroPSO. R package version 0.5-1 . doi:10.5281/zenodo.1287350. [p376]

Viviane Philipps
Inserm, Bordeaux Population Health Research Center, UMR 1219,
Univ. Bordeaux, F-33000 Bordeaux, France
146 rue Léo Saignat
33076 Bordeaux Cedex
France
E-mail: viviane.philipps@u-bordeaux.fr

Boris P. Hejblum
Inserm, Bordeaux Population Health Research Center, UMR 1219,
Inria BSO SISTM,
Vaccine Research Institute (VRI), F-94000 Créteil, France
Univ. Bordeaux, F-33000 Bordeaux, France
146 rue Léo Saignat
33076 Bordeaux Cedex
France
E-mail: boris.hejblum@u-bordeaux.fr

Mélanie Prague
Inserm, Bordeaux Population Health Research Center, UMR 1219,
Inria BSO SISTM,
Vaccine Research Institute (VRI), F-94000 Créteil, France
Univ. Bordeaux, F-33000 Bordeaux, France
146 rue Léo Saignat
33076 Bordeaux Cedex
France
E-mail: melanie.prague@inria.fr

Daniel Commenges
Inserm, Bordeaux Population Health Research Center, UMR 1219,
Inria BSO SISTM,
Univ. Bordeaux, F-33000 Bordeaux, France
146 rue Léo Saignat
33076 Bordeaux Cedex
France
E-mail: daniel.commenges@u-bordeaux.fr

Cécile Proust-Lima
Inserm, Bordeaux Population Health Research Center, UMR 1219,
Univ. Bordeaux, F-33000 Bordeaux, France
146 rue Léo Saignat
33076 Bordeaux Cedex
France
E-mail: cecile.proust-lima@inserm.fr

DRHotNet: An R package for detecting differential risk hotspots on a linear network

by Álvaro Briz-Redón, Francisco Martínez-Ruiz and Francisco Montes

Abstract One of the most common applications of spatial data analysis is detecting zones, at a certain scale, where a point-referenced event under study is especially concentrated. The detection of such zones, which are usually referred to as hotspots, is essential in certain fields such as criminology, epidemiology, or traffic safety. Traditionally, hotspot detection procedures have been developed over areal units of analysis. Although working at this spatial scale can be suitable enough for many research or practical purposes, detecting hotspots at a more accurate level (for instance, at the road segment level) may be more convenient sometimes. Furthermore, it is typical that hotspot detection procedures are entirely focused on the determination of zones where an event is (overall) highly concentrated. It is less common, by far, that such procedures focus on detecting zones where a specific type of event is overrepresented in comparison with the other types observed, which have been denoted as differential risk hotspots. The R package **DRHotNet** provides several functionalities to facilitate the detection of differential risk hotspots within a linear network. In this paper, **DRHotNet** is depicted, and its usage in the R console is shown through a detailed analysis of a crime dataset.

Introduction

Hotspot detection consists of finding zones across space where a certain event is highly concentrated. There exists a wide variety of methods in the literature that allow researchers to identify hotspots at a certain level of accuracy or spatial aggregation. Some of them have been massively used in the last decades, including certain local indicators of spatial association such as LISA (Anselin, 1995) or the Getis-Ord statistic (Getis and Ord, 1992), and the spatial scan statistic (Kulldorff, 1997). The first two of these methods are implemented in the R package **spdep** (Bivand et al., 2013), whereas the scan statistic is implemented in **DCluster** (Gómez-Rubio et al., 2005) (although other R packages also provide an implementation of these methods). Furthermore, many new R packages focused on hotspot detection have been released in the last few years. Most of them are model-based and oriented to disease mapping studies that are carried out over administrative (areal) units (Allévius, 2018; Gómez-Rubio et al., 2019; Meyer et al., 2017).

However, the analysis of certain types of events requires detecting hotspots at a level of spatial accuracy greater than that provided by administrative or regular areal units. Indeed, many research studies of the fields of criminology (Andresen et al., 2017; Weisburd, 2015) and traffic safety (Briz-Redón et al., 2019b; Nie et al., 2015; Xie and Yan, 2013) that have been published in recent years were entirely carried out on road network structures rather than on administrative units. More specifically, some quantitative criminologists have estimated that around 60% of the total variability in crime incidence occurs at the street segment level (Schnell et al., 2017; Steenbeek and Weisburd, 2016). A fact that shows the essentiality of using road segments instead of areal structures to properly capture the spatial concentration of certain events.

Fortunately, road network structures were introduced in the context of spatial statistics some years ago, providing the basis for analyzing events lying on such structures, which are usually referred to as linear networks. Indeed, a planar linear network, L, is defined as a finite collection of line segments, $L = \cup_{i=1}^{n} l_i$, in which each segment contains the points $l_i = [u_i, v_i] = \{tu_i + (1-t)v_i : t \in [0,1]\}$ (Ang et al., 2012; Baddeley et al., 2015, 2017). Following graph theory nomenclature, these segments are sometimes referred to as the edges of the linear network, whereas the points that determine the extremes of such segments are known as the vertices of the network.

Hence, a point process $\boldsymbol{X}$ on L is a finite point process in the plane (Diggle, 2013) such that all points of $\boldsymbol{X}$ lie on the network L, whereas a collection of events that is observed on L is known as a point pattern, $\boldsymbol{x}$, on L (Ang et al., 2012; Baddeley et al., 2015, 2017). In particular, when every event of a point pattern has one or several attributes, the point pattern is referred to as a marked point pattern. Each of the attributes, which can be in the form of either a numerical or a categorical variable, is known as a mark of the pattern. The intensity function of a process on L, denoted by $\lambda(x)$, satisfies that the number $n(\boldsymbol{X} \cap B)$ of points of $\boldsymbol{X}$ falling in $B \subset L$ has expectation $\mathbb{E}[n(\boldsymbol{X} \cap B)] = \int_B \lambda(u) \mathrm{d}_1 u$, where $\mathrm{d}_1 u$ denotes integration with respect to arc length (McSwiggan et al., 2017). Thus, given a point pattern, a typical objective is to estimate the intensity function of the process from which the locations of the events are assumed to be drawn, either through parametric or nonparametric techniques (Diggle, 2013).

The investigation of spatial patterns lying on linear networks has gained attention in the last few years. The design of new and more accurate/efficient kernel density estimators (McSwiggan et al., 2017; Moradi et al., 2018, 2019; Rakshit et al., 2019b), the introduction of graph-related intensity measures (Eckardt and Mateu, 2018), the construction of local indicators of spatial association (Eckardt and Mateu, 2021), or the estimation of relative risks (McSwiggan et al., 2019) are some topics that have recently started to be developed for linear networks.

Besides the theoretical advances, it is worth noting that using linear networks for carrying out a spatial or spatio-temporal analysis entails certain technical difficulties. In this regard, the R package **spatstat.linnet** of the **spatstat** family (Baddeley et al., 2015) provides multiple specific functions that allow R users to carry out statistical analyses on linear networks. Furthermore, efforts are constantly being made to reduce the computational cost of adapting certain classical spatial techniques to the singular case of linear networks (Rakshit et al., 2019a).

Despite the existing necessity of analyzing point-referenced data coming from certain fields of research at the street level, there are not many software tools fully designed for hotspot detection on road networks. One relevant contribution in this regard is the KDE+ software (Bíl et al., 2016), but this is not integrated into R. The package **DRHotNet** is specifically prepared for allowing R users to detect differential risk hotspots (zones where a type of event is overrepresented) along a linear network. While the function `relrisk.lpp` from **spatstat.linnet** also allows R users to compute the spatially-varying probability of a type of event located in a linear network, **DRHotNet** provides a full procedure to detect, with high accuracy, the segments of a network where the probability of occurrence of a given type of event is considerably higher and optimize hotspot detection in terms of a prediction accuracy index widely used in criminology and other fields. There is no other R package currently providing this functionality.

A procedure for detecting differential risk hotspots

The procedure for differential risk hotspot detection available in **DRHotNet** was introduced by Briz-Redón et al. (2019a), who also show an application of the method considering a traffic accident dataset. Overall, hotspot detection methods can be classified into partition-, hierarchy- and density-based methods (Deng et al., 2019). The one implemented in **DRHotNet** belongs to the last of these three groups.

Hence, the following subsections describe each of the steps that are carried out by the **DRHotNet** package. The specification and exemplification of the functions required for each of them are given in the following sections.

Estimating a relative probability surface

The first step consists in using kernel density estimation to infer the relative probability of occurrence for a certain type of event along a linear network. Given a marked point pattern $\boldsymbol{x} = \{x_1, ..., x_n\}$, where a binary mark y_i indicates if x_i corresponds, or not, to the type of event of interest, the following expression is used to estimate this relative probability (Kelsall and Diggle, 1998):

$$p_h(x) = \frac{\lambda_h^{\{x_i:y_i=1\}}(x)}{\lambda_h^{\{x_1,...,x_n\}}(x)}, \tag{1}$$

where $\lambda_h^{\{x_i:y_i=1\}}(x)$ is an estimate of the intensity of the pattern formed by the events of interest (those that satisfy $y_i = 1$) at location x, and $\lambda_h^{\{x_1,...,x_n\}}(x)$ is an estimate of the intensity of the complete pattern at x. Both estimates are nonparametric and depend on a bandwidth parameter, h. Specifically, in **DRHotNet** package, $\lambda_h^{\{x_i:y_i=1\}}(x)$ and $\lambda_h^{\{x_1,...,x_n\}}(x)$ are computed according to the network-constrained equal-split continuous kernel density estimation provided by McSwiggan et al. (2017). This version of kernel density is implemented in the function `density.lpp` of **spatstat.linnet**, which computes kernel density values rapidly by solving the classical heat equation defined on the linear network, as shown by McSwiggan et al. (2017). Despite the great performance of this approach, the computational cost of this version of kernel smoothing increases quadratically with h, so choosing very high values of h can become too time-consuming.

Therefore, Equation 1 allows establishing a relative probability surface (referring to the type of event represented by $y_i = 1$) on the linear network. To this end, given a location, x, of the linear network, the events which mainly contribute to the estimation of $p_h(x)$ are those situated within a linear radius (following the linear network structure) of h meters from x. Thus, increasing the value of

h leads to smoother representations of the probability surface, whereas smaller values of the bandwidth parameter produce the opposite effect. On the choice of an optimal bandwidth parameter, the method described by McSwiggan et al. (2019) could be followed, which modifies previous proposals made for planar patterns (Kelsall and Diggle, 1995a,b).

Estimating a relative probability surface implies, in practice, estimating a relative probability value at the middle points of the segments forming the road network. In this regard, it is necessary to split the original road network structure into shorter line segments called lixels (Xie and Yan, 2008), which are analogous to pixels for a planar surface. This step provides accuracy and homogeneity to the process of hotspot formation. Then, the segments satisfying certain conditions (details are provided in the next section) are selected and joined together forming the differential risk hotspots.

Determining differential risk hotspots

Once a relative probability surface has been estimated along the linear network, it is time to detect differential risk hotspots. The procedure for their detection relies on two parameters k and n. Parameter k is used together with the standard deviation of all the relative probabilities estimated to define a threshold that needs to be exceeded to consider a segment as a candidate to be part of a differential risk hotspot. Then, parameter n is employed to select the segments, from those satisfying the threshold condition, with n or more events at a distance lower than h. More precisely, a segment, i, of the linear network is considered to be part of a differential risk hotspot if:

$$\hat{p}_i > mean(\{\hat{p}_j\}_{j=1}^{S}) + k \cdot SD(\{\hat{p}_j\}_{j=1}^{S})$$

$$\#\{x \in \{x_1, ..., x_n\} : d_L(x, m_i) < h\} \geq n,$$

where $\hat{p}_i$ is the relative probability of occurrence for the type of event of interest estimated at the middle point of segment i, *mean* and *SD* denote, respectively, the mean and the standard deviation of a finite set of numbers, S is the number of segments of the linear network, # denotes the cardinality of a finite set, m_i is the middle point of segment i, and $d_L(x, m_i)$ represents the shortest-path distance (distance along the network) between x and m_i.

The segments that fulfill the two conditions indicated above are then joined, forming differential risk hotspots. More precisely, two segments satisfying the aforementioned conditions belong to the same differential risk hotspot if they are neighbors. Given two segments of a linear network, a neighboring relationship exists between them if they share a vertex of the network, which is equivalent to the *queen* criterion used for defining polygon neighborhoods (Lloyd, 2010). This relationship between segments is referred to as a first-order one. Similarly, higher-order neighboring relationships are defined recursively (for instance, for a second-order relationship): i and j are second-order neighbors if one neighbor of i and one neighbor of j are first-order neighbors.

Note that choosing a higher value of either k or n implies being stricter in terms of determining differential risk segments. If the choice is too strict, segments that meet both conditions may not be found. One criterion for finding suitable values of k and n is to perform a sensitivity analysis on k and n (consisting of testing a set of plausible values for k and n) and choosing a combination of both parameters that maximizes a measure that reflects the ability of the procedure to capture the overrepresentation of the event of interest along the network. These issues are clarified in subsequent sections.

Measuring differential risk hotspots importance and significance

The prediction accuracy index

Given a collection of hotspots in a planar surface, Chainey et al. (2008) defined the prediction accuracy index (PAI) as follows:

$$\text{PAI} = \frac{\text{Hit rate}}{\text{Area percentage}} = \frac{n/N}{a/A}, \tag{2}$$

where n is the total number of events that lie on the hotspots, N is the total number of events observed, a is the area formed by all the hotspots together, and A is the total area of the surface. Hence, a higher value of PAI is desirable in terms of hotspot detection as it represents a higher concentration of the events in relation to the size of the space they cover. This formula can be easily adapted to the case of linear networks using segment lengths in the place of areas.

In the context of detecting hotspots where the relative probability of one type of event is particularly high, the PAI needs to be modified. A type-specific version of the PAI (denoted by PAI$_t$) was proposed in Briz-Redón et al. (2019a) (for linear networks):

$$\text{PAI}_t = \frac{n_t/N_t}{\ell/L}, \tag{3}$$

where n_t is the number of events of the type of interest that lie on the hotspots, N_t is the total number of events of that type that are observed, ℓ is the length of all the hotspots together, and L is the total length of the network. The interpretation of the PAI$_t$ is analogous to the one for the PAI. That is, a higher PAI$_t$ value indicates a higher proportion of the type of interest compared to the proportion of road network length. Some researchers have recently explored the possibility of using the number of segments (in total and within the hotspots) instead of segment length (Drawve and Wooditch, 2019) in the denominator of this formula, but we opted for segment length proportions.

It is also worth noting that the PAI$_t$ can be computed for the whole set of differential risk hotspots detected, as indicated above, or for each differential risk hotspot individually. The first option is useful for comparing the level of spatial concentration that two or more types of events show along the network or for assessing the efficiency of different hotspot detection procedures. The second option is suitable for determining which differential risk hotspot maximizes the quotient between the proportion of the type and the proportion of road length spanned, which may represent the importance of the hotspot.

Estimating a *p*-value

Finally, assigning a statistical significance value to each differential risk hotspot is necessary to avoid the possibility of focusing on certain microzones of the network that do not deserve such attention. Thus, a Monte Carlo approach was used to estimate an empirical p-value for each differential risk hotspot yielded by the previous step of the procedure. This approach is similar in spirit to the one proposed by Bíl et al. (2013), which is applied in the KDE+ software mentioned before.

Getting back to previous notation, if $\{x_1, ..., x_n\}$ is a marked point pattern, and y_i a binary mark that indicates if x_i corresponds, or not, to the type of event under analysis, the Monte Carlo approach implemented consists in generating K simulated datasets where the locations are left fixed and the marks permutated. Concretely, this means to keep the locations $\{x_1, ..., x_n\}$ and to obtain a new collection of marks defined by $y_i^k = y_{\rho(i)}$, where k indicates the iteration number ($k = 1, ..., K$), and ρ is a permutation of the first n natural numbers. For each simulation, the average relative probability presented by each differential risk hotspot (computed as the weighted average per segment length of the relative probabilities estimated at the middle points of the segments composing the hotspot) is obtained and denoted by $\hat{s}_k$ ($k = 1, ..., K$). Therefore, if $\hat{r}$ represents such average relative probability for a hotspot, considering the original dataset, the rank of $\hat{r}$ within the ordered vector formed by the $\hat{s}_k$'s and $\hat{r}$ allows estimating an empirical p-value for the corresponding hotspot:

$$p = 1 - \frac{\#\{k \in \{1, ..., K\} : \hat{s}_k \leq \hat{r}\}}{K+1}$$

Dealing with linear networks in R

Classes and functions

Linear networks can be represented in R by the class `SpatialLines` of package **sp** (Pebesma and Bivand, 2005; Bivand et al., 2013) or by simple features with packages **sf** (Pebesma, 2018) and **sfnetworks** (van der Meer et al., 2021). However, the class `linnet` from the **spatstat.linnet** package is optimal for doing spatial analysis and modeling on linear networks.

There are several functions in R that facilitate the conversion between `SpatialLines` and `linnet` objects. Specifically, `as.linnet.SpatialLines` from the **maptools** (Bivand and Lewin-Koh, 2017) R package converts `SpatialLines` into `linnet` objects, whereas the double application (in this order) of the `as.psp` and `as.SpatialLines.psp` functions of the **spatstat.geom** and **maptools** packages, respectively, enable the conversion from a `linnet` object into a `SpatialLines` one.

Other useful functions available in **spatstat.linnet** for the use of linear networks are, for example, `connected.linnet` (computes the connected components of a linear network), `diameter.linnet` (computes the diameter and bounding radius of a linear network), `insertVertices` (inserts new vertices in a linear network), and `thinNetwork` (removes vertices or segments from a linear network).

Preprocessing the linear network

Although not strictly necessary, preprocessing the linear network is convenient to facilitate the subsequent statistical analysis. The **SpNetPrep** package (Briz-Redón, 2019) can be used for this purpose. The manual edition of the network, the addition of directionality, and the curation of a point pattern lying on a network can be performed through a Shiny-based application implemented in this package if the linear network represents a road network (which is the most common scenario and the one we assume for the example shown in this paper). Furthermore, **SpNetPrep** contains a function, `SimplifyLinearNetwork`, that allows users to reduce the network's complexity by merging some pairs of edges of the network that meet certain conditions on their length and angle. In this regard, another option is to use the `gSimplify` function of **rgeos** (Bivand and Rundel, 2020), which provides an implementation of the Douglas-Peuker algorithm for curve simplification (Douglas and Peucker, 1973).

Creating a point pattern on a linear network

Function `lpp` of package **spatstat.linnet** can be used to create an R object that represents a point pattern lying on a linear network. The `lpp` function only requires the coordinates of the events and a `linnet` object corresponding to a linear network. For instance, the following commands can be typed to create a point pattern of 100 points over the `simplenet` network provided by **spatstat.data**, which lies within the $[0,1] \times [0,1]$ window:

```
> x <- runif(100, 0, 1)
> y <- runif(100, 0, 1)
> simplenet.lpp <- lpp(data.frame(x, y), simplenet)
```

Marks can be attached to the points forming the pattern by introducing several more columns next to the `x` and `y` coordinates. For example, one can introduce a continuous random mark following a standard normal distribution or a categorical random mark.

```
> random_cont_mark <- rnorm(100, 0, 1)
> random_cat_mark <- letters[round(runif(100, 0, 5))+1]
> simplenet.lpp <- lpp(data.frame(x, y, random_cont_mark, random_cat_mark),
                                  simplenet)
```

In order to fit the objective of computing a relative probability for one type of event, categorical marks are required. However, recoding a continuous mark into several categories to facilitate the estimation of a specific relative risk is one possible alternative.

Using DRHotNet

This section shows the complete use of the **DRHotNet** package with a dataset of crime events recorded in Chicago (Illinois, US). The reader should note that some of the outputs may vary slightly due to possible changes over time in some of the packages used. First of all, the following R libraries have to be loaded to reproduce the example:

```
> library(DRHotNet)
> library(lubridate)
> library(maptools)
> library(raster)
> library(rgeos)
> library(sp)
> library(spatstat.core)
> library(spatstat.linnet)
> library(spdep)
> library(SpNetPrep)
> library(tigris)
```

Downloading and preparing the linear network

The examples provided in this section fully employ open data available for Chicago. First, geographic data from Chicago was downloaded from the United States Census Bureau through package **tigris** (Walker, 2016). Specifically, census tracts and the road network of the state of Massachusetts were loaded into the R console. The function `intersect` from the package **raster** (Hijmans, 2019) can be used.

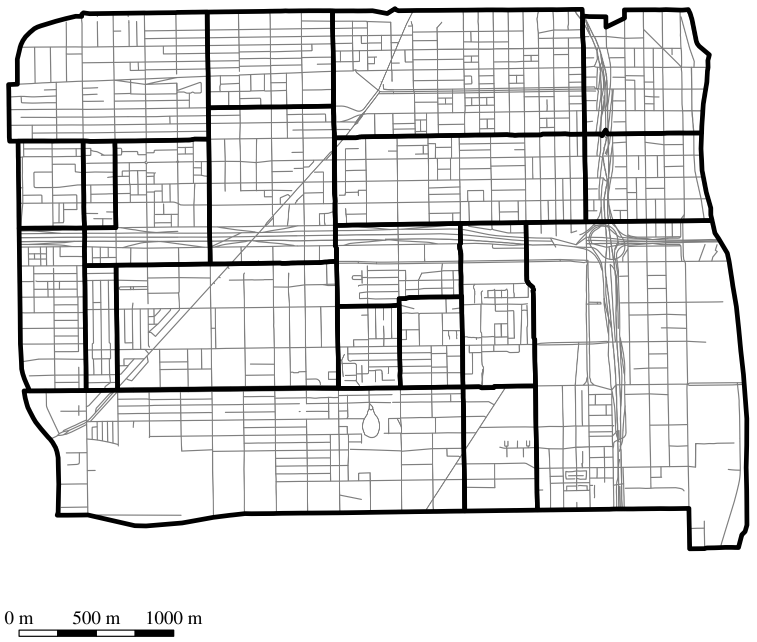


Figure 1: Road network (in gray) corresponding to the Near West Side Community Area of Chicago. Census tracts of the area are overlayed in black.

```
> cook.tracts <- tracts(state = "Illinois", county = "031", class = "sp")
> class(cook.tracts)
[1] "SpatialPolygonsDataFrame"
attr(,"package")
[1] "sp"
> cook.network <- roads(state = "Illinois", county = "031", year = 2011,
                        class = "sp")
> class(cook.network)
[1] "SpatialLinesDataFrame"
attr(,"package")
[1] "sp"
```

The objects `cook.tracts` and `cook.network` are composed of 1319 polygons and 86227 lines, respectively (as of the end of June 2021). Now, both spatial objects are intersected to construct a smaller road network that corresponds to the Near West Side Community Area of Chicago.

```
> names.tracts <- as.character(cook.tracts@data[,"NAME"])
> select.tracts <- c("8378","2804","8330","2801","2808","2809","8380",
                     "8381","8331","2819","2827","2828","8382","8329",
                     "2831","2832","8333","8419","8429","2838")
> cook.tracts.select <- cook.tracts[which(names.tracts%in%select.tracts),]
> chicago.SpLines <- intersect(cook.network, cook.tracts.select)
> length(chicago.SpLines)
[1] 1357
```

Object `chicago.SpLine` (`SpatialLinesDataFrame`) has 1357 lines. Then, this object's coordinates are converted into UTM (Chicago's UTM zone is 16):

```
> chicago.SpLines <- spTransform(chicago.SpLines,
                                 "+proj=utm +zone=16 ellps=WGS84")
```

Now the corresponding `linnet` object is created:

```
> chicago.linnet <- as.linnet(chicago.SpLines)
> chicago.linnet
Linear network with 10602 vertices and 11910 lines
Enclosing window:
    rectangle = [442564.6, 447320] x [4634170, 4637660] units
```

It is worth noting how the transformation of the network into a `linnet` object increases dramatically the number of line segments (from 1357 to 11910). This is a consequence of the fact that `SpatialLines` objects can handle curvilinear segments, made of multiple line segments, as a single line. However, `linnet` objects follow the definition of the linear network provided in the Introduction section, which excludes this possibility.

It is required that the network is fully connected to allow the computation of a distance between any pair of points. This can be checked with the function `connected`.

```
> table(connected(chicago.linnet, what = "labels"))

    1     2     3     4     5     6     7     8     9
10575     5     2     2     2     5     2     2     7
```

This output indicates that there is a connected component of 10575 vertices and other eight components with only a few vertices. The use of `connected` with the option `what = "components"` enables us to extract the larger connected component for the analysis, discarding the others.

```
> chicago.linnet.components <- connected(chicago.linnet,
                                          what = "components")
> chicago.linnet.components[[1]]
Linear network with 10575 vertices and 11891 lines
Enclosing window:
rectangle = [442564.6, 447320] x [4634170, 4637660]
units
> chicago.linnet <- chicago.linnet.components[[1]]
```

At this point, it is worth considering the possibility of reducing the network's complexity. The function `SimplifyLinearNetwork` of **SpNetPrep** can be used for this purpose. A reasonable choice of the parameters is `Angle = 20` and `Length = 50` (Briz-Redón, 2019). This choice of the parameters means that a pair of segments meeting at a second-degree vertex are merged into one single segment if the angle they form (measured from $0°$ to $90°$) is lower than $20°$ and if the length of each of them is lower than 50 m. Hence, the network's complexity is reduced (in terms of the number of segments and lines) while its geometry is preserved. The following lines of code execute `SimplifyLinearNetwork` and redefine `chicago.SpLine` according to the structure of the final `linnet` object.

```
> chicago.linnet <- SimplifyLinearNetwork(chicago.linnet,
                                           Angle = 20, Length = 50)
> chicago.linnet
Linear network with 3026 vertices and 4339 lines
Enclosing window:
rectangle = [442564.6, 447320] x [4634170, 4637660] units
> chicago.SpLines <- as.SpatialLines.psp(as.psp(chicago.linnet))
```

Thus, the final road network from Chicago that we use for the analysis has 4339 lines and 3026 vertices. An example of how `SimplifyLinearNetwork` reduces the network's complexity is shown in Figure 2, which corresponds to a squared zone of Chicago's network with a diameter of 600 m.

Downloading and preparing crime data

Point-referenced crime datasets corresponding to several cities from the United States of America can be downloaded through the R package **crimedata** (Ashby, 2019). Concretely, **crimedata** currently provides (as of June 2021) crime open data recorded in Austin, Boston, Chicago, Detroit, Fort Worth, Kansas City, Los Angeles, Louisville, Mesa, New York, San Francisco, Tucson, and Virginia Beach. Therefore, the function `get_crime_data` from this package can be used for downloading a dataset of crime events recorded in Chicago in the period 2007-2018.

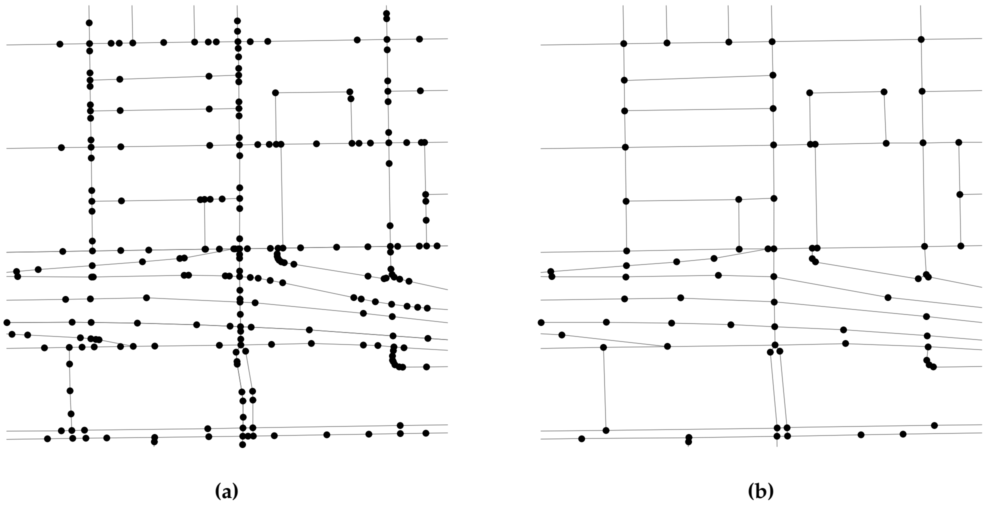


Figure 2: Extracting a part of the road network structure analyzed from Chicago. Original structure extracted (left), made of 285 lines and 272 vertices, and simplified version of it (right) with 122 lines and 109 vertices.

```
> chicago.crimes <- get_crime_data(years = 2007:2018, cities = "Chicago")
> dim(chicago.crimes)
[1] 39206    12
```

The year, month, and hour of occurrence of each crime can be extracted with the corresponding functions of the package **lubridate** (Grolemund and Wickham, 2011).

```
> chicago.crimes$year <- year(chicago.crimes$date_single)
> chicago.crimes$month <- month(chicago.crimes$date_single)
> chicago.crimes$hour <- hour(chicago.crimes$date_single)
```

Then, a marked point pattern lying on `chicago.linnet` can be created with function `lpp` to provide the framework required by the **DRHotNet** package. A `data.frame` is passed to `lpp` including the coordinates of the events (in UTM), the type of event according to the receiver of the offense (`offense_against`), and the year, month, and hour of occurrence that have been just computed.

```
> chicago.crimes.coord <- data.frame(x = chicago.crimes$longitude,
                                     y = chicago.crimes$latitude)
> coordinates(chicago.crimes.coord) <-~ x + y
> lonlat_proj <- "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
> utm_proj <- "+proj=utm +zone=16 ellps=WGS84"
> proj4string(chicago.crimes.coord) <- lonlat_proj
> chicago.crimes.coord <- spTransform(chicago.crimes.coord, utm_proj)
> X.chicago <- lpp(data.frame(x = chicago.crimes.coord@coords[,1],
                   y = chicago.crimes.coord@coords[,2],
                   offense_against = chicago.crimes$offense_against,
                   year = chicago.crimes$year,
                   month = chicago.crimes$month,
                   hour = chicago.crimes$hour),
                   chicago.linnet)
Warning message:
38006 points were rejected as lying outside the specified window
```

A total of 38006 points are rejected because they lie outside the road network. Hence, a marked point pattern of 1200 crimes that lie on `chicago.linnet` remains for the analysis. The four marks are categorical, presenting the following values and absolute frequencies:

```
> table(X.chicago$data$offense_against)
other  persons property  society
```

```
   86       257       749       108
> table(X.chicago$data$year)
2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018
 126  131   97  111  100  110   87   84   76   93   92   93
> table(X.chicago$data$month)
  1   2   3   4   5   6   7   8   9  10  11  12
103  71 101  92 110 100 128 102 111 109  95  78
> table(X.chicago$data$hour)
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
45 34 29 25 19 14 18 41 53 51 48 54 71 71 69 71 60 72
18 19 20 21 22 23
64 67 60 52 62 50
```

Hence, **DRHotNet** functionalities are now applicable to `X.chicago`. In the main example shown, the relative occurrence of crimes against persons along the road network included in `X.chicago` is analyzed. To this end, the functions of the package are used following the steps that have been established for the differential risk hotspot detection methodology previously described.

Estimating a relative probability surface

The function `relpnet` of **DRHotNet** has to be used at first to estimate a relative probability surface over the linear network. As it has been explained, this implies estimating a relative probability in the middle point of each segment of the network.

```
> rel_probs_persons <- relpnet(X = X.chicago,
                               lixel_length = 50,
                               h = 250,
                               mark = "offense_against",
                               category_mark = "persons")
```

The available parameter `finespacing` is by default set to `FALSE` in order to reduce the computational cost (as `FALSE` is the default value, it can be omitted in the specification of the function). If `finespacing` is set to `TRUE`, more accurate kernel density estimates are obtained at short segments, but the computational cost can become prohibitive. The interested reader can consult the documentation of the `densityHeat` function of **spatstat.linnet** to know precisely how this parameter works.

In this example, an upper bound of 50 m is chosen for the lixel length. This means that segments shorter than 50 m are not split, whereas those longer than 50 m are split into several shorter lixels of no more than 50 m lengths. This operation is performed internally by `relpnet` with the function `lixellate` from **spatstat.linnet**. The bandwidth parameter, h, is set to 250 m. The `mark` and `category_mark` parameters are used to specify the type of event that is under analysis.

The exploration of the object `rel_probs_persons` with function `str` allows the user to check that the choice of a 50 m threshold for the lixel length produces 8617 segments along the network. Thus, a relative probability is estimated in the middle point of each of these segments, which can be accessed by typing `$probs`:

```
> str(rel_probs_persons)
List of 5
 $ probs        : num [1:8617] 0.129 0.138 0.202 0.492 0.662 ...
 $ lixel_length : num 50
 $ h        : num 250
 $ mark         : chr "offense_against"
 $ category_mark: chr "persons"
```

The function `plotrelp` can then be used to obtain a map of the relative probability surface like the one shown in Figure 3b. Figure 3 also contains the relative probability surfaces corresponding to the choices of $h = 125$ (Figure 3a), $h = 500$ (Figure 3c) and $h = 1000$ (Figure 3d). In this particular case, using an Intel(R) Core(TM) i7-6700HQ processor, it takes approximately 1 second to execute `relpnet` if $h = 125$, and 8 seconds if $h = 1000$.

Figure 1 allows us to observe that the choice of a larger value for h smooths the relative probability surface, which in the case of $h = 500$ or $h = 1000$ leads to the configuration of a small number of clearly distinguishable zones of the network in terms of the relative probability of offenses against persons. Indeed, whereas the use of $h = 125$ allows the user to obtain quite extreme relative probability values

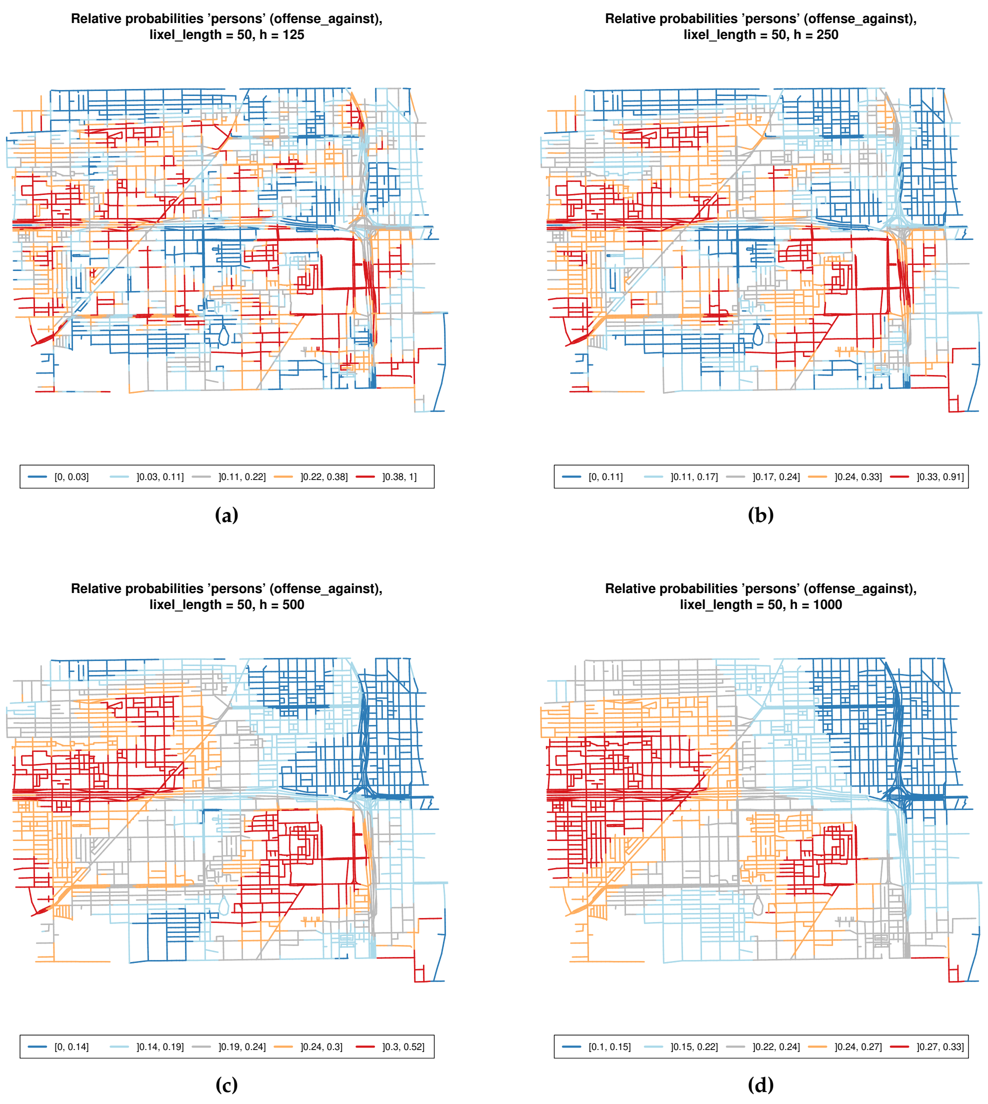

Relative probabilities 'persons' (offense_against), lixel_length = 50, h = 125
[0, 0.03]
]0.03, 0.11]
]0.11, 0.22]
]0.22, 0.38]
]0.38, 1]
(a)
Relative probabilities 'persons' (offense_against), lixel_length = 50, h = 250
[0, 0.11]
]0.11, 0.17]
]0.17, 0.24]
]0.24, 0.33]
]0.33, 0.91]
(b)
Relative probabilities 'persons' (offense_against), lixel_length = 50, h = 500
[0, 0.14]
]0.14, 0.19]
]0.19, 0.24]
]0.24, 0.3]
]0.3, 0.52]
(c)
Relative probabilities 'persons' (offense_against), lixel_length = 50, h = 1000
[0.1, 0.15]
]0.15, 0.22]
]0.22, 0.24]
]0.24, 0.27]
]0.27, 0.33]
(d)

Figure 3: Outputs from the function `plotrelp` for the following choices of h: 125 (a), 250 (b), 500 (c) and 1000 (d).

at some segments of the network (the relative probability estimates cover the [0,1] interval), choosing $h = 1000$ causes that all the relative probabilities lie in the interval $[0.10, 0.33]$.

In view of Figure 3, we consider that $h = 250$ is a reasonable choice (although some procedures for bandwidth selection could be explored for taking this decision). Therefore, this selection of the bandwidth parameter is maintained to display now how the `drhot` function of the package works.

Detecting differential risk hotspots

The function `drhot` needs four parameters to be specified: `X` (a point pattern on a linear network), `rel_probs` (an object like the one obtained in the previous step), `k`, and `n`. Parameters `k` and `n` control the differential risk hotspot procedure, as it has been explained before. For example, we can try with `k = 1` and `n = 30`:

```
> hotspots_persons <- drhot(X = X.chicago,
                            rel_probs = rel_probs_persons,
                            k = 1, n = 30)
```

The output of the function `drhot` presents the following structure:

```
> str(hotspots_persons)
List of 8
```

```
 $ DRHotspots   :List of 5
  ..$ : num [1:42] 248 249 502 503 576 ...
  ..$ : num 2971
  ..$ : num 7551
 $ k            : num 1
 $ n            : num 30
 $ lixel_length : num 50
 $ h        : num 250
 $ mark         : chr "offense_against"
 $ category_mark: chr "persons"
 $ PAI_t     : num 12.1
```

The first component of hotspots_persons contains the differential risk hotspots that have been detected for the values of k and n provided to drhot. In this case, three differential risk hotspots are found, which are formed by 42, 1, and 1 road segments, respectively. The object hotspots_persons also contains the values of the parameters involved in the computation of the hotspots and a final component that includes the global PAI$_t$ for the set, which is 12.10 in this example.

The function drsummary can be used to provide a summary of each of the differential risk hotspots determined by drhot:

```
> summary_persons <- drsummary(X = X.chicago,
                               rel_prob = rel_probs_persons,
                               hotspots = hotspots_persons)
```

The output of drsummary includes a count of the number of events located within each differential risk hotspot and how many of these correspond to the category "persons":

```
> summary_persons[,c("Events type (ctr)", "All events (ctr)",
                     "Prop. (ctr)")]
  Events type (ctr) All events (ctr) Prop. (ctr)
1                16               35        0.46
2                 0                0         NaN
3                 0                0         NaN
```

The summary also contains the length (in meters) of each differential risk hotspot and the PAI$_t$ that corresponds to each of them:

```
> summary_persons[,c("Length in m (ctr)", "PAI_t (ctr)")]
  Length in m (ctr) PAI_t (ctr)
1           1532.51       12.59
2             42.99        0.00
3             25.22        0.00
```

Furthermore, the output of drsummary also provides the same statistics for an extension of each of the hotspots. The reason to include this information is that if there are not many events available in the dataset (as it happens in this example), the method can determine differential risk hotspots where very few events, if any, have taken place. Indeed, in the output of drsummary shown above, there are two hotspots including zero events. The fact of employing kernel density estimation to infer a relative probability surface makes it more convenient to think of each differential risk hotspot as the union of a center or core (what drhot returns, the hotspot itself) and an extension of it. Hence, by considering an extension of the differential risk hotspot, one can better appreciate the zone of the network that has been accounted for in the estimation of the relative probability values corresponding to the segments of the hotspot.

By default, the extension computed by drsummary coincides with a neighbourhood of the segments forming each differential risk hotspot of order $o = \frac{h}{\text{Lixel length}}$ (rounded to the nearest integer), although a different order can be specified through the parameter order_extension. In this example, we have $o = \frac{250}{50} = 5$, which is used by drsummary if no other order is indicated:

```
> summary_persons[,c("Events type (ext)", "All events (ext)",
                     "Prop. (ext)")]
  Events type (ext) All events (ext) Prop. (ext)
1                22               57        0.39
2                11               26        0.42
3                10               24        0.42
> summary_persons[,c("Length in m (ext)", "PAI_t (ext)")]
```

```
  Length in m (ext) PAI_t (ext)
1           6283.36        4.22
2           1700.86        7.80
3           1189.04       10.14
```

It can be observed that all extensions of the differential risk hotspots include a reasonable number of events and that the corresponding proportions of offenses against persons are clearly above the global proportion for the dataset, which is $257/1200 = 21.42$.

Assessing the statistical significance of the hotspots

Following the choice of `k = 1` and `n = 30`, it only remains to estimate a p-value for each differential risk hotspot detected. This can be done by calling the function `drsummary` again and specifying `compute_p_value = T`. A total of 20 iterations are selected for performing the Monte Carlo simulation process:

```
> summary_persons <- drsummary(X = X.chicago,
                               rel_prob = rel_probs_persons,
                               hotspots = hotspots_persons,
                               compute_p_value = T,
                               n_it = 200)
> summary_persons$`p-value`
[1] 0.000 0.000 0.000
```

Therefore, the five differential risk hotspots detected with $k = 1$ and $n = 30$ are statistically significant ($p < 0.05$). It is worth noting, however, that the usual significance level of 0.05 should be reduced (corrected) if many differential risk hotspots are detected to avoid the presence of multiple comparison problems.

Choosing k and n

Remember that a higher value of either `k` or `n` represents using a more stringent criterion regarding hotspot detection. This is illustrated through the four maps available in Figure 4, which can be generated with the function `plothot` of **DRHotNet**. For instance, as in the following example for the object `hotspots` created previously (which corresponds to Figure 4d):

```
> plothot(X = X.chicago, hotspots = hotspots_persons)
```

Indeed, Figure 4 shows the differential risk hotspots that `drhot` detects for the choices of `k = 0.5` and `n = 20` (Figure 4a), `k = 1.5` and `n = 20` (Figure 4b), `k = 1` and `n = 10` (Figure 4c), and `k = 1` and `n = 30` (Figure 4d). Two of these combinations of conditions on k and n are implicitly represented by the other two. Consequently, the differential risk hotspots shown in Figure 4b are contained in Figure 4a, and those in Figure 4d are contained in Figure 4c. The choice of `k = 1` and `n = 30` leads to the highest global PAI$_t$ among the four combinations of parameters indicated with the value of 12.1 mentioned before. In this regard, we recommend performing a sensitivity analysis on the values of k and n to decide which combination is more convenient. The sensitivity analysis carried out by Briz-Redón et al. (2019a) yielded that a choice around $k = 1.5$ and $n = 45$ was optimal in terms of the PAI$_t$ for the traffic accident dataset that was investigated. However, each dataset should require a specific analysis.

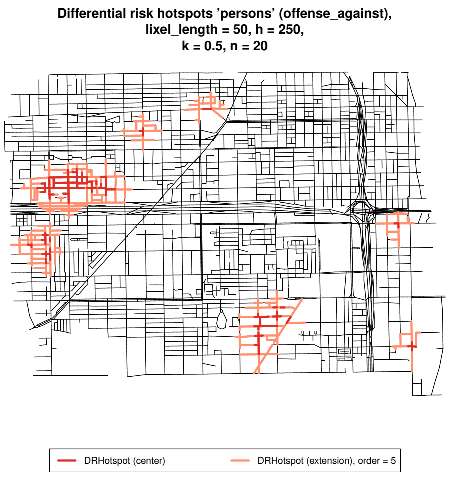
Thus, a sensitivity analysis on k and n can be carried out with the function `drsens`. The user has to provide a point pattern (`X`), an object containing the relative probabilities of a type of event along the network (`rel_probs`) and a set of values for k and n (`ks` and `ns`, respectively):

```
> sensitivity_analysis <- drsens(X = X.chicago,
                                 rel_prob = rel_probs_persons,
                                 ks = seq(0,3,0.5),
                                 ns = seq(10,30,5))
> sensitivity_analysis
        n = 10 n = 15 n = 20 n = 25 n = 30
k = 0     3.34   5.07   6.75  12.35  12.05
k = 0.5   4.47   6.52   7.92  12.82  12.05
k = 1     5.15   8.62   8.92  10.93  12.05
k = 1.5   5.94   9.12  10.89  10.88  13.70
k = 2     2.16  10.06     NA     NA     NA
```

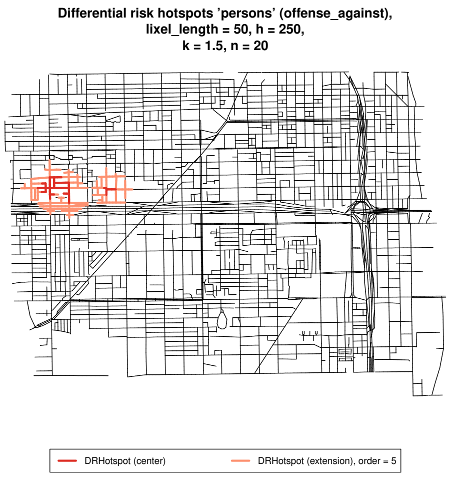
```
k = 2.5     NA      NA      NA      NA      NA
k = 3       NA      NA      NA      NA      NA
```

The output from `drsens` is a matrix that contains the PAI$_t$ values that correspond to each combination of k and n indicated by `ks` and `ns`. A `NA` value represents that no differential risk hotspots can be found for such a combination of parameters. According to this matrix, the highest PAI$_t$ value is achieved for $k = 1.5$ and $n = 30$.

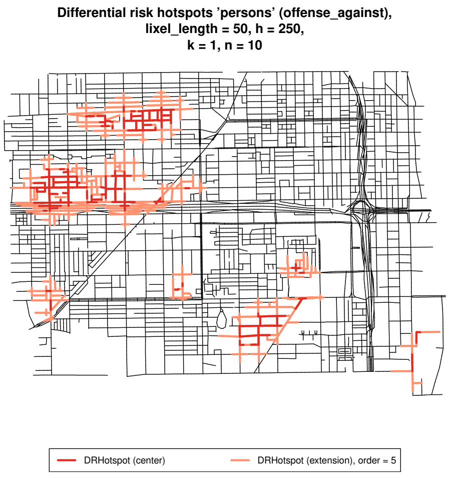
Therefore, one can choose the values of k and n that maximize the PAI$_t$ (considering the parameters provided in `ks` and `ns`) to determine the final set of differential risk hotspots. However, this criteria may sometimes lead to detecting a very low number of differential risks hotspots and hence to miss zones of the network that may also deserve some attention. Hence, a more conservative approach could be considering several combinations of k and n that yield some of the highest values of PAI$_t$ and explore each set of differential risk hotspots associated. Then, one could investigate the output of `drsummary` for each combination of parameters (including the computation of p-values) to better decide which zones of the network are relevant for the type of event of interest.


Differential risk hotspots 'persons' (offense_against),
lixel_length = 50, h = 250,
k = 0.5, n = 20
DRHotspot (center)
DRHotspot (extension), order = 5

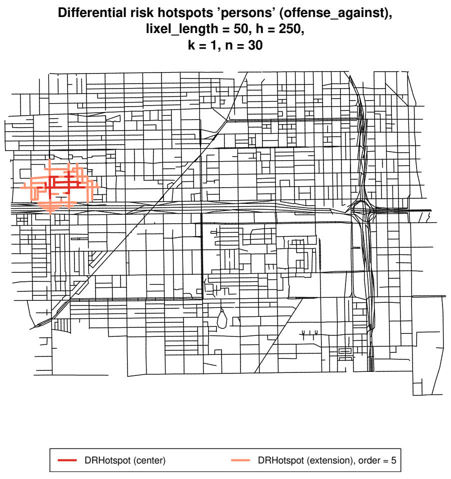
(a)


Differential risk hotspots 'persons' (offense_against),
lixel_length = 50, h = 250,
k = 1.5, n = 20
DRHotspot (center)
DRHotspot (extension), order = 5

(b)


Differential risk hotspots 'persons' (offense_against),
lixel_length = 50, h = 250,
k = 1, n = 10
DRHotspot (center)
DRHotspot (extension), order = 5

(c)


Differential risk hotspots 'persons' (offense_against),
lixel_length = 50, h = 250,
k = 1, n = 30
DRHotspot (center)
DRHotspot (extension), order = 5

(d)

Figure 4: Outputs from the function `plothot` for the following choices of k and n: $k = 0.5$ and $n = 20$ (a), $k = 1.5$ and $n = 20$ (b), $k = 1$ and $n = 10$ (c), and $k = 1$ and $n = 30$ (d).

Other applications of the methodology

This final section shows the results that are obtained for other type of events for comparative purposes. First, the marks `X.chicago` are recoded into binary outcomes as follows:

```
> year_after_2012 <- ifelse(X.chicago$data$year>2012, "Yes", "No")
> month_winter <- ifelse(X.chicago$data$month%in%c(12,1,2), "Yes", "No")
> hour_21_3 <- ifelse(X.chicago$data$hour%in%c(21:23,0:3), "Yes", "No")
> marks(X.chicago) <- data.frame(as.data.frame(marks(X.chicago)),
                                 year_after_2012 = year_after_2012,
                                 month_winter = month_winter,
                                 hour_21_3 = hour_21_3)
```

The relative probability surfaces have to be computed. We used the same values for `lixel.length` and `h` than in the previous examples.

```
> rel_probs_after_2012 <- relpnet(X = X.chicago,
                                  lixel_length = 50,
                                  h = 250,
                                  mark = "year_after_2012",
                                  category_mark = "Yes")
> rel_probs_winter <- relpnet(X = X.chicago,
                              lixel_length = 50,
                              h = 250,
                              mark = "month_winter",
                              category_mark = "Yes")
> rel_probs_21_3 <- relpnet(X = X.chicago,
                            lixel_length = 50,
                            h = 250,
                            mark = "hour_21_3",
                            category_mark = "Yes")
```

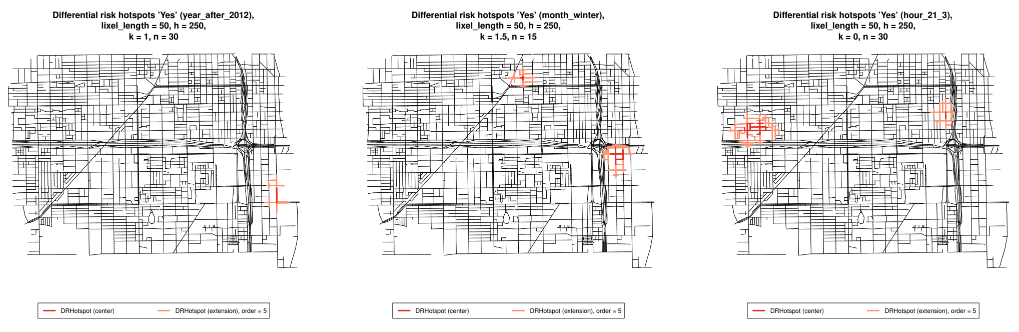
The corresponding sensitivity analyses are carried out:

```
> sensitivity_after_2012 <- drsens(X = X.chicago,
                                   rel_prob = rel_probs_after_2012,
                                   ks = seq(0,3,0.5),
                                   ns = seq(10,30,5))
> sensitivity_after_2012
        n = 10 n = 15 n = 20 n = 25 n = 30
k = 0     2.47   3.35   4.59   6.36  10.96
k = 0.5   3.08   4.28   7.11  14.00  15.09
k = 1     2.84   4.38   8.33  25.76  29.98
k = 1.5   2.65   4.63   7.03   0.00   0.00
k = 2     2.24   0.00     NA     NA     NA
k = 2.5     NA     NA     NA     NA     NA
k = 3       NA     NA     NA     NA     NA
> sensitivity_winter <- drsens(X = X.chicago,
                               rel_prob = rel_probs_winter,
                               ks = seq(0,3,0.5),
                               ns = seq(10,30,5))
> sensitivity_winter
        n = 10 n = 15 n = 20 n = 25 n = 30
k = 0     2.75   3.63   4.81   5.17   4.86
k = 0.5   2.83   3.60   4.41   8.35   0.00
k = 1     3.62   3.86   0.00   0.00     NA
k = 1.5   7.59   8.56   0.00     NA     NA
k = 2       NA     NA     NA     NA     NA
k = 2.5     NA     NA     NA     NA     NA
k = 3       NA     NA     NA     NA     NA
> sensitivity_21_3 <- drsens(X = X.chicago,
                             rel_prob = rel_probs_21_3,
                             ks = seq(0,3,0.5),
                             ns = seq(10,30,5))
> sensitivity_21_3
        n = 10 n = 15 n = 20 n = 25 n = 30
k = 0     2.86   4.55   5.41   6.75   7.78
k = 0.5   3.50   5.21   6.02   7.11   7.34
k = 1     4.17   5.90   4.84   0.00   0.00
k = 1.5   4.78   6.92   5.52     NA     NA
```

```
k = 2      5.14   5.64   3.00      NA      NA
k = 2.5    4.59   1.75   0.00      NA      NA
k = 3        NA     NA     NA      NA      NA
```

The highest PAI$_t$ values for `year_after_2012`, `month_winter`, and `hour_21_3` are 29.98, 8.56, and 7.78, respectively. The differential risk hotspots that are obtained for the combination of k and n that lead to these PAI$_t$ values can be visualized with `plothot` (Figure 5):

```
> hotspots_after_2012 <- drhot(X = X.chicago,
                               rel_prob = rel_probs_after_2012,
                               k = 1,
                               n = 30)
> plothot(X = X.chicago, hotspots_after_2012)

> hotspots_winter <- drhot(X = X.chicago,
                           rel_prob = rel_probs_winter,
                           k = 1.5,
                           n = 15)
> plothot(X = X.chicago, hotspots_winter)

> hotspots_21_3 <- drhot(X = X.chicago,
                         rel_prob = rel_probs_21_3,
                         k = 0,
                         n = 30)
> plothot(X = X.chicago, hotspots_21_3)
```


Differential risk hotspots 'Yes' (year_after_2012), lixel_length = 50, h = 250, k = 1, n = 30
DRHotspot (center)
DRHotspot (extension), order = 5
Differential risk hotspots 'Yes' (month_winter), lixel_length = 50, h = 250, k = 1.5, n = 15
DRHotspot (center)
DRHotspot (extension), order = 5
Differential risk hotspots 'Yes' (hour_21_3), lixel_length = 50, h = 250, k = 0, n = 30
DRHotspot (center)
DRHotspot (extension), order = 5

(a) (b) (c)

Figure 5: Outputs from the function `plothot` for the marks `year_after_2012`, `month_winter`, and `hour_21_3` and the categorical value `Yes` for the three, considering the combinations of k and n that maximize the PAI$_t$ (for the values of k and n tested).

Summary

The R package **DRHotNet** for detecting differential risk hotspots on linear networks has been described. The use of linear networks in the context of hotspot detection is becoming more important over the years, particularly in the fields of criminology and traffic safety. In addition, it is also of great interest sometimes to detect zones of a linear network where a certain type of event is especially overrepresented. Hence, **DRHotNet** consists of an easy-to-use tool implemented in R to accurately locate the microzones of a linear network where the incidence of a type of event is considerably higher than in the rest of it.

Bibliography

B. Allévius. scanstatistics: Space-time anomaly detection using scan statistics. *The Journal of Open Source Software*, 3:515, 2018. [p380]

M. A. Andresen, A. S. Curman, and S. J. Linning. The trajectories of crime at places: understanding the patterns of disaggregated crime types. *Journal of Quantitative Criminology*, 33(3):427–449, 2017. [p380]

Q. W. Ang, A. Baddeley, and G. Nair. Geometrically corrected second order analysis of events on a linear network, with applications to ecology and criminology. *Scandinavian Journal of Statistics*, 39(4): 591–617, 2012. [p380]

L. Anselin. Local indicators of spatial association—LISA. *Geographical Analysis*, 27(2):93–115, 1995. [p380]

M. P. Ashby. Studying Crime and Place with the Crime Open Database: Social and Behavioural Scienes. *Research Data Journal for the Humanities and Social Sciences*, 1(aop):1–16, 2019. [p386]

A. Baddeley, E. Rubak, and R. Turner. *Spatial point patterns: methodology and applications with R*. CRC Press, 2015. [p380, 381]

A. Baddeley, G. Nair, S. Rakshit, and G. McSwiggan. "Stationary" point processes are uncommon on linear networks. *Stat*, 6(1):68–78, 2017. [p380]

M. Bíl, R. Andrášik, and Z. Janoška. Identification of hazardous road locations of traffic accidents by means of kernel density estimation and cluster significance evaluation. *Accident Analysis & Prevention*, 55:265–273, 2013. [p383]

M. Bíl, R. Andrášik, T. Svoboda, and J. Sedoník. The KDE+ software: a tool for effective identification and ranking of animal-vehicle collision hotspots along networks. *Landscape Ecology*, 31(2):231–237, 2016. [p381]

R. Bivand and N. Lewin-Koh. *maptools: Tools for Reading and Handling Spatial Objects*, 2017. URL `https://CRAN.R-project.org/package=maptools`. R package version 0.9-2. [p383]

R. Bivand and C. Rundel. *rgeos: Interface to Geometry Engine - Open Source ('GEOS')*, 2020. URL `https://CRAN.R-project.org/package=rgeos`. R package version 0.5-3. [p384]

R. S. Bivand, E. Pebesma, and V. Gómez-Rubio. *Applied spatial data analysis with R, Second edition*. Springer, NY, 2013. URL `http://www.asdar-book.org/`. [p380, 383]

Á. Briz-Redón. SpNetPrep: An R package using Shiny to facilitate spatial statistics on road networks. *Research Ideas and Outcomes*, 5:e33521, 2019. [p384, 386]

Á. Briz-Redón, F. Martínez-Ruiz, and F. Montes. Identification of differential risk hotspots for collision and vehicle type in a directed linear network. *Accident Analysis & Prevention*, 132:105278, 2019a. [p381, 383, 391]

Á. Briz-Redón, F. Martínez-Ruiz, and F. Montes. Spatial analysis of traffic accidents near and between road intersections in a directed linear network. *Accident Analysis & Prevention*, 132:105252, 2019b. [p380]

S. Chainey, L. Tompson, and S. Uhlig. The utility of hotspot mapping for predicting spatial patterns of crime. *Security Journal*, 21(1-2):4–28, 2008. [p382]

M. Deng, X. Yang, Y. Shi, J. Gong, Y. Liu, and H. Liu. A density-based approach for detecting network-constrained clusters in spatial point events. *International Journal of Geographical Information Science*, 33(3):466–488, 2019. [p381]

P. J. Diggle. *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. CRC press, 2013. [p380]

D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. [p384]

G. Drawve and A. Wooditch. A research note on the methodological and theoretical considerations for assessing crime forecasting accuracy with the predictive accuracy index. *Journal of Criminal Justice*, page 101625, 2019. [p383]

M. Eckardt and J. Mateu. Point patterns occurring on complex structures in space and space-time: An alternative network approach. *Journal of Computational and Graphical Statistics*, 27(2):312–322, 2018. [p381]

M. Eckardt and J. Mateu. Second-order and local characteristics of network intensity functions. *TEST*, 30(2):318–340, 2021. [p381]

A. Getis and J. Ord. The Analysis of Spatial Association by Use of Distance Statistics. *Geographical Analysis*, 24(3), 1992. [p380]

G. Grolemund and H. Wickham. Dates and Times Made Easy with lubridate. *Journal of Statistical Software*, 40(3):1–25, 2011. URL http://www.jstatsoft.org/v40/i03/. [p387]

V. Gómez-Rubio, J. Ferrándiz-Ferragud, and A. López-Quílez. Detecting clusters of disease with R. *Journal of Geographical Systems*, 7(2):189–206, 2005. [p380]

V. Gómez-Rubio, P. Moraga, J. Molitor, and B. Rowlingson. Dclusterm: Model-based detection of disease clusters. *Journal of Statistical Software, Articles*, 90(14):1–26, 2019. ISSN 1548-7660. doi: 10.18637/jss.v090.i14. URL https://www.jstatsoft.org/v090/i14. [p380]

R. J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2019. URL https://CRAN.R-project.org/package=raster. R package version 2.8-19. [p385]

J. E. Kelsall and P. J. Diggle. Kernel estimation of relative risk. *Bernoulli*, 1(1-2):3–16, 1995a. [p382]

J. E. Kelsall and P. J. Diggle. Non-parametric estimation of spatial variation in relative risk. *Statistics in Medicine*, 14(21-22):2335–2342, 1995b. [p382]

J. E. Kelsall and P. J. Diggle. Spatial variation in risk of disease: a nonparametric binary regression approach. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(4):559–573, 1998. [p381]

M. Kulldorff. A spatial scan statistic. *Communications in Statistics-Theory and Methods*, 26(6):1481–1496, 1997. [p380]

C. Lloyd. *Spatial data analysis: an introduction for GIS users*. Oxford University Press, 2010. [p382]

G. McSwiggan, A. Baddeley, and G. Nair. Kernel density estimation on a linear network. *Scandinavian Journal of Statistics*, 44(2):324–345, 2017. [p380, 381]

G. McSwiggan, A. Baddeley, and G. Nair. Estimation of relative risk for events on a linear network. *Statistics and Computing*, pages 1–16, 2019. [p381, 382]

S. Meyer, L. Held, and M. Höhle. Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance. *Journal of Statistical Software*, 77(11), 2017. [p380]

M. M. Moradi, F. J. Rodríguez-Cortés, and J. Mateu. On kernel-based intensity estimation of spatial point patterns on linear networks. *Journal of Computational and Graphical Statistics*, 27(2):302–311, 2018. [p381]

M. M. Moradi, O. Cronie, E. Rubak, R. Lachieze-Rey, J. Mateu, and A. Baddeley. Resample-smoothing of Voronoi intensity estimators. *Statistics and Computing*, 29(5):995–1010, 2019. [p381]

K. Nie, Z. Wang, Q. Du, F. Ren, and Q. Tian. A network-constrained integrated method for detecting spatial cluster and risk location of traffic crash: A case study from Wuhan, China. *Sustainability*, 7 (3):2662–2677, 2015. [p380]

E. Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 2018. URL https://journal.r-project.org/archive/2018/RJ-2018-009/index.html. [p383]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL https://CRAN.R-project.org/doc/Rnews/. [p383]

S. Rakshit, A. Baddeley, and G. Nair. Efficient Code for Second Order Analysis of Events on a Linear Network. *Journal of Statistical Software*, 90(1):1–37, 2019a. [p381]

S. Rakshit, T. Davies, M. M. Moradi, G. McSwiggan, G. Nair, J. Mateu, and A. Baddeley. Fast Kernel Smoothing of Point Patterns on a Large Network using Two-dimensional Convolution. *International Statistical Review*, 2019b. [p381]

C. Schnell, A. A. Braga, and E. L. Piza. The influence of community areas, neighborhood clusters, and street segments on the spatial variability of violent crime in Chicago. *Journal of Quantitative Criminology*, 33(3):469–496, 2017. [p380]

W. Steenbeek and D. Weisburd. Where the action is in crime? An examination of variability of crime across different spatial units in The Hague, 2001–2009. *Journal of Quantitative Criminology*, 32(3): 449–469, 2016. [p380]

L. van der Meer, L. Abad, A. Gilardi, and R. Lovelace. *sfnetworks: Tidy Geospatial Networks*, 2021. URL `https://CRAN.R-project.org/package=sfnetworks`. R package version 0.5.2. [p383]

K. Walker. tigris: An R package to access and work with geographic data from the US Census Bureau. *The R Journal*, 8(2):231–242, 2016. [p384]

D. Weisburd. The law of crime concentration and the criminology of place. *Criminology*, 53(2):133–157, 2015. [p380]

Z. Xie and J. Yan. Kernel density estimation of traffic accidents in a network space. *Computers, Environment and Urban Systems*, 32(5):396–406, 2008. [p382]

Z. Xie and J. Yan. Detecting traffic accident clusters with network kernel density estimation and local spatial statistics: an integrated approach. *Journal of Transport Geography*, 31:64–71, 2013. [p380]

Álvaro Briz-Redón
Statistics and Operations Research
University of València
Spain
`alvaro.briz@uv.es`

Francisco Martínez-Ruiz
Statistics Office
City Council of València
Spain
`fmartinezr@valencia.es`

Francisco Montes
Statistics and Operations Research
University of València
Spain
`francisco.montes@uv.es`

tramME: Mixed-Effects Transformation Models Using Template Model Builder

by Bálint Tamási and Torsten Hothorn

Abstract Linear transformation models constitute a general family of parametric regression models for discrete and continuous responses. To accommodate correlated responses, the model is extended by incorporating mixed effects. This article presents the R package **tramME**, which builds on existing implementations of transformation models (**mlt** and **tram** packages) as well as Laplace approximation and automatic differentiation (using the **TMB** package), to calculate estimates and perform likelihood inference in mixed-effects transformation models. The resulting framework can be readily applied to a wide range of regression problems with grouped data structures.

Introduction

Datasets with grouped observations are abundant in the applied statistical practice. Clustering, hierarchical designs, longitudinal studies, or repeated measurements can all lead to grouped data structures. The common property of these datasets is that observations within groups, defined by one or more grouping factors, cannot be treated as independent. In order to draw a valid inference, the statistical model has to address the issue of correlated observations. Mixed-effects models represent one of the main approaches dealing with this type of regression problem. In this approach, the observations are assumed to be independent *conditionally* on a set of random effects that aim to capture unmodeled group-level heterogeneity. The reader is referred, for example, to the textbook by Demidenko (2013) for an exposition and examples of the usage of mixed-effects models. Several R packages exist that implement mixed-effects models for specific types of regression problems. The two most notable examples are **nlme** by Pinheiro et al. (2021) and **lme4** by Bates et al. (2015) for linear, non-linear, and generalized linear mixed-effects models, respectively.

Linear transformation models aim to directly specify the conditional distribution function of an outcome variable in a regression setting. Hothorn (2020) proposed a fully parametric approach using a flexible monotone increasing transformation function that is estimated from the data. The resulting general model family can be applied to a wide range of problems with at least ordered discrete outcome variables. In fact, many of the popular regression models can be expressed as special cases of the linear transformation model framework. Most recently, Tian et al. (2020) reviewed the approach followed in this study and compared it to an alternative semiparametric formulation using extensive simulations. By introducing random effects in the linear transformation model, it becomes applicable in a very diverse set of regression problems where the observations are correlated due to repeated measurements or grouped designs.

The structure of this article is as follows: After a brief, and somewhat technical, introduction of the methodology and the implementation in Section 2.2, Section 2.3 demonstrates, through a series of examples, how the package **tramME** (Tamási and Hothorn, 2021) can be applied to estimate regression models with various response types and data structures. Finally, Section 2.4 discusses a few issues concerning the implementation of our model.

Mixed-effects transformation models

The model class in the R package **tramME** is an extension of the transformation model approach described by Hothorn et al. (2018) and implemented in the R packages **mlt** and **tram** by Hothorn (2020) and Hothorn and Barbanti (2021), respectively. These resources provide an introduction to fully parameterized transformation models for independent observations.

Formally, we are interested in models that parameterize the conditional distribution function directly,

$$\mathbb{P}\left(Y \le y \mid \mathbf{x}, \mathbf{u}, \gamma\right) = F_Z\left(h(y;\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma\right) \qquad \gamma \sim \mathcal{N}_q(\mathbf{0}, \boldsymbol{\Sigma}), \tag{1}$$

where F_Z denotes a pre-specified error distribution function (or inverse-link function), which is monotone increasing and maps from the real numbers to the closed interval $[0,1]$. Typically, F_Z is set to the CDF of a simple continuous distribution, hence the name "error distribution". The baseline transformation function is $h(y;\boldsymbol{\vartheta})$, which is also a monotonic increasing function parameterized with the vector $\boldsymbol{\vartheta}$. For the fixed and random effects design matrices, respectively, $\mathbf{x}^\top$ and $\mathbf{u}^\top$ are suitable row vectors. The vector $\boldsymbol{\beta}$ contains the fixed effects, while γ comprises of the stacked (possibly multiple) random effects. The distribution of the random effects is assumed to be multivariate Gaussian with zero mean and covariance matrix $\boldsymbol{\Sigma}$, which typically has a sparse block structure.

As Table 1 shows, specific choices of the error distribution and the baseline transformation function lead to different types of regression models. In the R package **tramME**, seven main model types are distinguished, mainly based on the class of their outcome variable. Moreover, the functions `SurvregME()` and `PolrME()` allow to specify multiple error distributions or baseline transformations and hence increasing the number of available model types.

Function	Name	F_Z	$h(y;\boldsymbol{\vartheta})$
`LmME()`	Mixed-effects normal linear regression	Standard Gaussian	Linear basis
`BoxCoxME()`	Non-normal (Box-Cox-type) linear mixed-effects regression	Standard Gaussian	Bernstein basis
`ColrME()`	Mixed-effects continuous outcome logistic regression	Standard logistic	Bernstein basis
`CoxphME()`	Mixed-effects parametric Cox regression	Minimum extreme value	Bernstein basis
`SurvregME()`	Mixed-effects parametric survival models	Multiple options	Multiple options
`PolrME()`	Mixed-effects regression models for ordinal outcomes	Multiple options	Discrete basis
`LehmannME()`	Mixed-effects Lehmann-alternative linear regression	Maximum extreme value	Bernstein basis

Table 1: Model types implemented in the **tramME** package. F_Z denotes the error distribution and the column $h(y;\boldsymbol{\vartheta})$ lists the basis functions the baseline transformation function utilizes.

As the table indicates, some of the models specify their transformation functions as general smooth functions, approximated with the use of polynomials in Bernstein form. The function $h(y;\boldsymbol{\vartheta})$ has to be monotonic increasing so that the conditional distribution function is also increasing. When using a set of order p polynomials in Bernstein form for the approximation of a general function, this restriction conveniently translates to the parameter restriction $\vartheta_i \leq \vartheta_{i+1}$ for all $i = 0, \ldots, p-1$.

The observations are assumed to be *conditionally independent*, and hence the likelihood has the form

$$\begin{aligned}\mathcal{L}\left(\boldsymbol{\vartheta},\boldsymbol{\beta},\boldsymbol{\Sigma}\right) &= \int_{\mathbb{R}^q} \mathcal{L}(\boldsymbol{\vartheta},\boldsymbol{\beta},\boldsymbol{\Sigma},\gamma)\,\mathsf{d}\gamma \\ &= \int_{\mathbb{R}^q} \prod_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\vartheta},\boldsymbol{\beta} \mid \gamma)\phi(\gamma;\boldsymbol{\Sigma})\,\mathsf{d}\gamma, \qquad (2)\end{aligned}$$

where $\mathcal{L}(\boldsymbol{\vartheta},\boldsymbol{\beta},\boldsymbol{\Sigma},\gamma)$ is the joint likelihood function, given the all observations, and $\mathcal{L}_i(\boldsymbol{\vartheta},\boldsymbol{\beta} \mid \gamma)$ denotes the individual *conditional* likelihood contributions. $\phi(\gamma;\boldsymbol{\Sigma})$ stands for the probability density function of the multivariate normal distribution with zero mean vector and covariance matrix $\boldsymbol{\Sigma}$. This latter function can be factorized further according to the covariance structure of the random effects.

One of the main advantages of working directly with the distribution function of the outcome is that it is simple to introduce (random) censoring and truncation in the estimation procedure. The conditional likelihood contributions under different types of censoring can be written as

$$\mathcal{L}_i(\boldsymbol{\vartheta},\boldsymbol{\beta} \mid \gamma) = \begin{cases} f_Z(h(y;\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma)h'(y;\boldsymbol{\vartheta}) & y \in \Xi \quad \text{“exact continuous”} \\ 1 - F_Z(h(\underline{y};\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma) & y \in (\underline{y},\infty) \cap \Xi \quad \text{“right-censored”} \\ F_Z(h(\bar{y};\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma) & y \in (-\infty,\bar{y}] \cap \Xi \quad \text{“left-censored”} \\ F_Z(h(\bar{y};\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma) \\ \qquad - F_Z(h(\underline{y};\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma) & y \in (\underline{y},\bar{y}] \cap \Xi \quad \text{“interval-censored”,} \end{cases}$$

where $f_Z()$ is the density function of the error distribution, $h'(y;\boldsymbol{\vartheta})$ is the first derivative of the baseline transformation function with respect to y, and Ξ denotes the sample space of Y.

The multidimensional integral in Equation (2), in general, does not have an analytical solution, but its value can be approximated using numerical methods. The package **tramME** applies the Laplace approximation to this problem, which relies on the quadratic Taylor expansion of the corresponding joint log-likelihood function.

The maximization of the logarithm of the likelihood function with respect to $\boldsymbol{\vartheta}$, $\boldsymbol{\beta}$, and $\boldsymbol{\Sigma}$, under a set of suitable constraints on $\boldsymbol{\vartheta}$ to make $h(y;\boldsymbol{\vartheta})$ monotone increasing, results in the maximum likelihood estimates of the model parameters. Standard likelihood theory, utilizing the ability to evaluate the log-likelihood function, the score function, and the Hessian, provides a basis for asymptotic inference in this family of models; see Hothorn et al. (2018) for more details on likelihood inference in transformation models.

The maximum likelihood estimation in **tramME** is done using the **TMB** package by Kristensen et al. (2016). The Template Model Builder (**TMB**) allows the user to define and estimate general, non-linear mixed-effects models. It was built on well-tested and high-performance C++ libraries, which results in a flexible yet efficient framework for estimating mixed models with possibly complex random effects structures; see, for example, Brooks et al. (2017) for performance comparisons in the context of the package **glmmTMB**. In **tramME**, **TMB** is used to evaluate the integral in Equation (2), using Laplace's method, and to calculate the derivatives of the log-likelihood function using automatic (or algorithmic) differentiation.

Applications

In this section, several applications of the transformation mixed models are presented, and wherever it is possible, also compared to other existing implementations. The examples shown here are by no means intended as complete analyses. They demonstrate how mixed-effects transformation models can be used in a broad range of regression problems and showcase the most important features implemented in the package **tramME**.

In each application, a simple version of a transformation mixed model is compared to the same model implemented by a benchmark package first. In a second step, extensions to more complex models not available other packages are fitted using package **tramME**. The R code illustrates similarities and differences in the user interfaces. The two model outputs allow a direct comparison of the model-agnostic implementation in **tramME** to the model-specific implementation in the benchmark package. The package **tramME** is, however, *not* intended as a replacement for well-tested implementations of important special cases of mixed models, such as linear mixed models in **lme4**, but as a tool for extending these implementations to more complex model variants.

Normal linear mixed model

As a first example, we model the average reaction times to a specific task from a sleep deprivation study described in Belenky et al. (2003). Figure 1 presents the reaction times against days of sleep deprivation for each of the 18 participants.

In this first example, we model the distribution of the average reaction time using random intercepts and random slopes for the effects of days of sleep deprivation.

$$\mathbb{P}\left(\texttt{Reaction} \le y \mid \texttt{Days}, \alpha_i, \beta_i\right) = \Phi\left(\vartheta_1 + \vartheta_2 y - \beta\texttt{Days} - \gamma_{1i} - \gamma_{2i}\texttt{Days}\right) \tag{3}$$
$$\begin{pmatrix}\gamma_{1i}\\ \gamma_{2i}\end{pmatrix} \sim \mathcal{N}_2\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\tau_1^2 & \tau_{12}\\ \tau_{12} & \tau_2^2\end{pmatrix}\right\}$$

Note that when the transformation function is assumed to be linear in the outcome variable, i.e., $h(y) = \vartheta_1 + \vartheta_2 y$, we arrive at a re-parameterized version of the normal linear mixed effects-model, and hence the results from `tramME::LmME()` are directly comparable to estimates using other mixed-effects regression packages such as **lme4**. Estimating the normal linear model with the **tramME**:

```
R> library("tramME")
R> sleep_lmME <- LmME(Reaction ~ Days + (Days | Subject), data = sleepstudy)
R> logLik(sleep_lmME)

'log Lik.' -876 (df=6)
```

To make the results from **lme4** comparable to the previous results, we set `REML = FALSE`, as the transformation mixed model implementation only supports the maximum likelihood estimation of

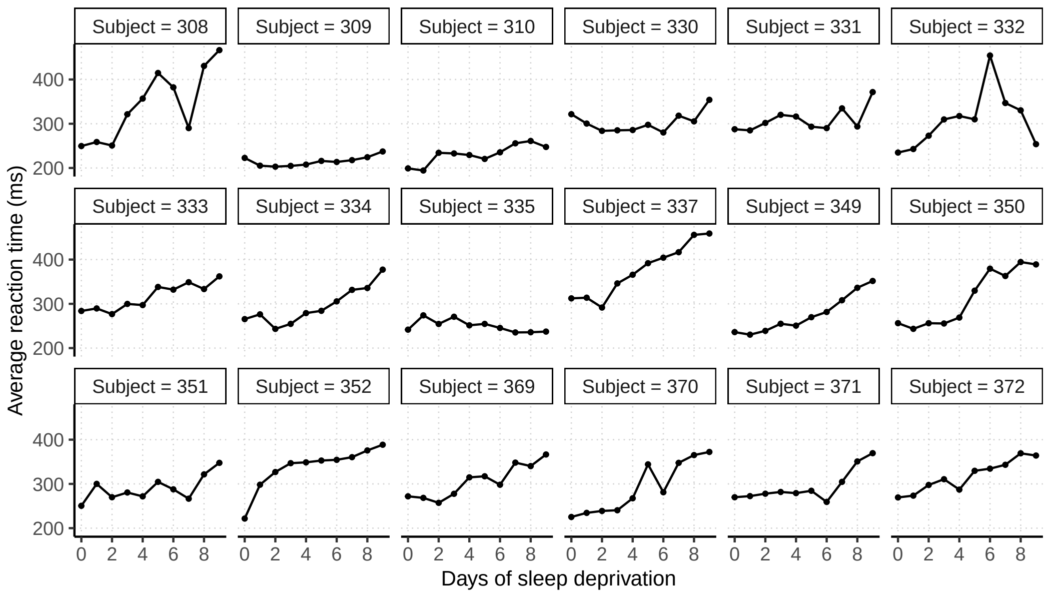


Figure 1: Sleep deprivation study: Average reaction times to a specific task of 18 participants after several days of sleep deprivation reported by Belenky et al. (2003).

the normal linear model specification.

```
R> library("lme4")
R> sleep_lmer <- lmer(Reaction ~ Days + (Days | Subject), data = sleepstudy,
+                     REML = FALSE)
R> logLik(sleep_lmer)

'log Lik.' -876 (df=6)
```

The as.lm = TRUE option of various methods in **tramME** facilitates the comparisons between the transformation model parameterization and the results of a linear mixed model parameterization. Coefficient estimates and their standard errors from the transformation model approach are

```
R> cbind(coef = coef(sleep_lmME, as.lm = TRUE),
+        se = sqrt(diag(vcov(sleep_lmME, as.lm = TRUE, pargroup = "fixef"))))

             coef   se
(Intercept) 251.4 6.63
Days         10.5 1.50
```

while the results from lmer are

```
R> summary(sleep_lmer)$coefficients

            Estimate Std. Error t value
(Intercept)    251.4       6.63   37.91
Days            10.5       1.50    6.97
```

Similarly, the standard deviations and correlations of the random effects and the standard deviations of the error terms are essentially the same

```
R> VarCorr(sleep_lmME, as.lm = TRUE) ## random effects


Grouping factor: Subject (18 levels)
Standard deviation:
(Intercept)        Days
      23.80        5.72


Correlations:
     (Intercept)
Days      0.0813
```

```
R> sigma(sleep_lmME) ## residual SD
```

```
[1] 25.6
```

```
R> VarCorr(sleep_lmer)
```

```
 Groups   Name        Std.Dev. Corr
 Subject  (Intercept) 23.78
          Days         5.72    0.08
 Residual             25.59
```

With the predict method of **tramME**, we can evaluate the fitted conditional distribution of the outcome on a scale specified by the user. Additionally, by setting type = "quantile", we can calculate the quantiles of the conditional distribution of the response.

```
R> ## Update to specify the support
R> sleep_lmME1b <- update(sleep_lmME, support = c(150, 520))
R> ## Set up grid to calculate conditional quantiles
R> nd <- expand.grid(Days = seq(min(sleepstudy$Days), max(sleepstudy$Days),
+                                length.out = 200),
+                    Subject = unique(sleepstudy$Subject))
R> ## The quantiles we want to calculate
R> pr <- c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975)
R> ## Specify the random effects values as predicted by the model
R> re <- ranef(sleep_lmME1b)
R> ## Calculate conditional quantiles
R> pred <- predict(sleep_lmME1b, newdata = nd, prob = pr, ranef = re,
+                  type = "quantile")
```

Note that in the code above, we first update the model and explicitly set the support of the outcome distribution. It is often helpful to define the support when we want to calculate the quantiles of the outcome distribution because in certain extreme cases, the calculated values may lie outside of the default support and, in these cases, predict.tramME (just as predict.mlt) will return censored values.

Because we are interested in conditional quantiles, we have to specify the values of the random effects on which we want to condition for each subject. In the example above, we simply set the predicted random effects values for the subjects of the sleepstudy dataset. Although it is relatively common in practice, one should be careful with using plug-in estimators of non-linear functions of the random effects (i.e., estimating these functions by evaluating at the point estimates of the random effects) as they can contain substantial bias (Thorson and Kristensen, 2016). To demonstrate how mixed-effects transformation models relax certain assumptions of the normal linear reference model and to showcase the functionality implemented in the **tramME** package, occasionally, we will rely on these estimators nevertheless.

Figure 2 presents the quantiles of the conditional distribution of reaction time from the model defined in Equation (3). The random intercepts and slopes capture separate time trends for each subject in the study. In the normal linear mixed model (estimated with LmME), the conditional quantiles are parallel lines. We will revisit this example when we relax certain assumptions of this initial model in Section 2.3.2.

tramME implements a version of score residuals that are defined as the score contributions of the individual observations with respect to an additional constant term that is fixed at zero.

$$\mathbb{P}\left(Y \le y \mid \mathbf{x}, \mathbf{u}, \gamma\right) = F_Z\left(h(y;\boldsymbol{\vartheta}) - \mathbf{x}^\top\boldsymbol{\beta} - \mathbf{u}^\top\gamma - \alpha_0\right) \qquad \gamma \sim \mathcal{N}_q(\mathbf{0}, \boldsymbol{\Sigma})$$
$$r_i = \left.\frac{\partial \ell_i(\boldsymbol{\vartheta}, \boldsymbol{\beta}, \boldsymbol{\Sigma}, \alpha_0)}{\partial \alpha_0}\right|_{\alpha_0 = 0,} \tag{4}$$

where $\ell_i(\boldsymbol{\vartheta}, \boldsymbol{\beta}, \boldsymbol{\Sigma}, \alpha_0)$ is the marginal log-likelihood contribution of observation i. It is straightforward to show that, in the case of the normal linear model, this is equal to the response residuals divided by the MLE of the error standard deviation. As previously, the comparison with the parameterization used by **lme4** is made easy by using the option as.lm = TRUE.

```
R> resid_lmME <- resid(sleep_lmME, as.lm = TRUE)
R> resid_lmer <- resid(sleep_lmer)
R> all.equal(resid_lmME, resid_lmer)
```

```
[1] "Mean relative difference: 8.04e-06"
```

Using the linear predictor of the mixed-effects transformation model, $(\mathbf{x}^\top\widehat{\boldsymbol{\beta}} + \mathbf{u}^\top\widehat{\gamma})$ calculated as

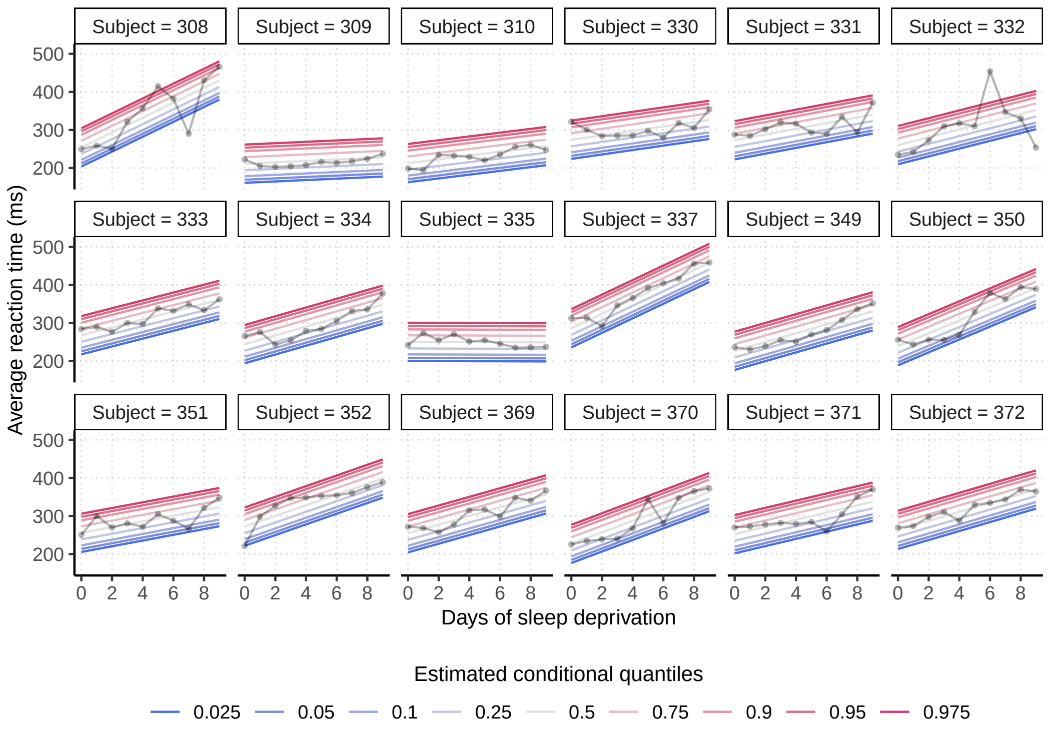


Figure 2: Quantiles of the conditional distribution of the outcome fitted to the `sleepstudy` data with the normal linear mixed-effects model (`LmME`).

```
R> lp <- predict(sleep_lmME, type = "lp")
```

we can construct plots for checking the residuals (Figure 3).

As the results of this section show, the transformation model approach implemented by `LmME()` leads to the same results as the maximum likelihood estimation of the traditional linear mixed model parameterization. The advantage of using the package **tramME** over other well-established implementations is that it can also be applied when classical model assumptions are not met. For the sleep deprivation experiments, the data analyst might wonder if assuming normal reaction times is appropriate and if clocking of reaction times was indeed as accurate as suggested by the data (milliseconds with four digits). The former issue requires a relaxation of the conditional normality assumption and the latter incorporation of interval-censoring in the likelihood. We will start with model estimation in the presence of interval-censored reaction times, which is outside the scope of **lme4**.

Let us assume that the measurement device used in the sleep deprivation study is only able to measure reaction times larger than 200 ms and only in 50 ms step sizes. If we want to take this reduced accuracy in the measurements into account, we have to deal with interval-censored observations as ignoring the censored nature of the outcomes could lead to biased parameter estimates.

With the following code, we create the interval-censored outcome vector using the `Surv` function of the **survival** package by Therneau (2021).

```
R> library("survival")
R> ub <- ceiling(sleepstudy$Reaction / 50) * 50
R> lb <- floor(sleepstudy$Reaction / 50) * 50
R> lb[ub == 200] <- 0
R> sleepstudy$Reaction_ic <- Surv(lb, ub, type = "interval2")
R> head(sleepstudy$Reaction_ic)

[1] [200, 250] [250, 300] [250, 300] [300, 350] [350, 400] [400, 450]
```

Using the interval-censored outcomes in the `LmME()`, function call will maximize the correct likelihood function.

```
R> sleep_lmME2 <- LmME(Reaction_ic ~ Days + (Days | Subject), data = sleepstudy)
R> logLik(sleep_lmME2)
```

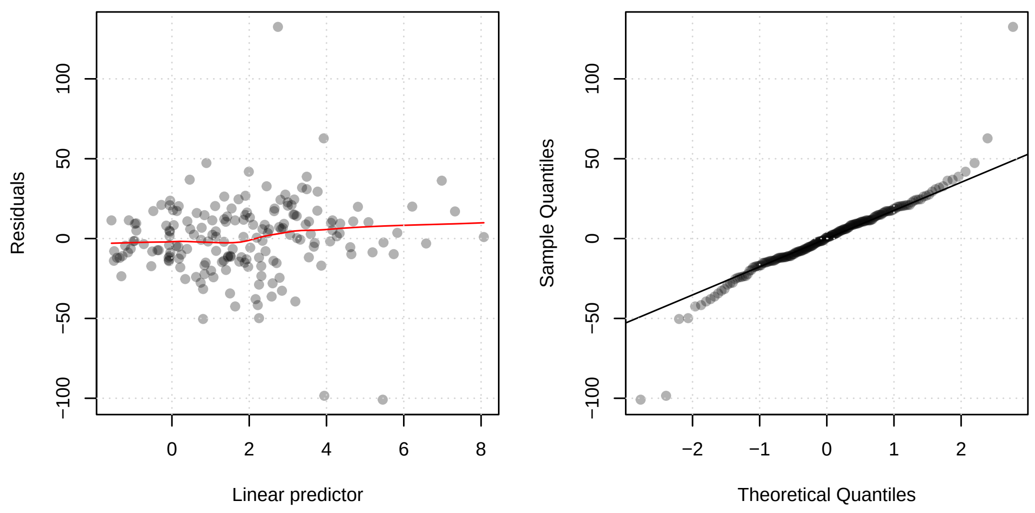


Figure 3: Residual plots of the normal linear mixed-effects transformation model fitted to the `sleepstudy` data. *Left:* Residuals plotted against the linear predictor. *Right:* QQ plot of the residuals against Gaussian quantiles.

```
'log Lik.' -201 (df=6)
```

The value of the log-likelihood is different because we are now calculating log-probabilities instead of log-densities of a continuous distribution. However, despite the decreased precision of the measurements, the parameter estimates are similar to what we got with the exactly observed outcomes.

```
R> cbind(coef = coef(sleep_lmME2, as.lm = TRUE),
+        se = sqrt(diag(vcov(sleep_lmME2, as.lm = TRUE, pargroup = "fixef"))))

             coef   se
(Intercept) 251.4 6.83
Days         10.5 1.62

R> sigma(sleep_lmME2)

[1] 28

R> VarCorr(sleep_lmME2, as.lm = TRUE)


Grouping factor: Subject (18 levels)
Standard deviation:
(Intercept)        Days
      22.30        5.94

Correlations:
     (Intercept)
Days      0.0536
```

The small estimated value of the correlation coefficient between the random slope and intercept suggests that a model with independent random effects might be more appropriate. To estimate such a model, we can use the same notation as in **lme4**

```
R> sleep_lmME3 <- LmME(Reaction_ic ~ Days + (Days || Subject), data = sleepstudy)
R> logLik(sleep_lmME3)

'log Lik.' -201 (df=5)
```

Comparing the two models using a likelihood ratio test, we see no evidence against the more parsimonious model (`sleep_lmME3`).

```
R> anova(sleep_lmME2, sleep_lmME3)
```

```
Model comparison

         Model 1: Reaction_ic ~ Days + (Days | Subject)
         Model 2: Reaction_ic ~ Days + (Days || Subject)

        npar logLik AIC BIC Chisq Chisq df Pr(>Chisq)
Model 2    5   -200 411 427
Model 1    6   -200 413 432  0.02        1       0.89
```

Note that the standard likelihood ratio tests provided by anova are very conservative for model comparisons that involve setting some of the random effects variances to zero due to the non-standard asymptotics of tests on the boundary of the parameter space (Self and Liang, 1987).

Box-Cox-type mixed-effects models

Substituting the linear baseline transformation function, $h(y) = \vartheta_1 + \vartheta_2 y$, with a general, monotonic increasing smooth function, we can relax the conditional normality assumption of the model discussed in Section 2.3.1. The transformation model approach proposed by Hothorn et al. (2018) uses Bernstein bases to approximate this general increasing function in a fully parametric manner, i.e., $h(y) = \mathbf{a}_{\text{Bs},K+1}(y)^\top \boldsymbol{\vartheta}$. The resulting model can be regarded as a version of the Box-Cox regression (Box and Cox, 1964), where the transformation of the response is estimated simultaneously with the model parameters. It should be pointed out that, although its approach is similar in spirit, **tramME** does not use the Box-Cox power transformation to approximate $h(y)$. For an implementation utilizing the original Box-Cox transform in the context of mixed-effects models, see the R package **boxcoxmix** by Almohaimeed and Einbeck (2020).

A more flexible version of the model described in (3) will take the form

$$\mathbb{P}\left(\texttt{Reaction} \le y \mid \texttt{Days}, \alpha_i, \beta_i\right) = \Phi\left(\mathbf{a}(y)^\top \boldsymbol{\vartheta} - \beta\texttt{Days} - \gamma_{1i} - \gamma_{2i}\texttt{Days}\right) \tag{5}$$

$$\begin{pmatrix}\gamma_{1i}\\ \gamma_{2i}\end{pmatrix} \sim \mathcal{N}_2\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\tau_1^2 & \tau_{12}\\ \tau_{12} & \tau_2^2\end{pmatrix}\right\}.$$

The Box-Cox-type transformation mixed model can be estimated using the BoxCoxME() function of the **tramME** package. For this specific application, we set the order of the polynomials in Bernstein form to 10.

```
R> sleep_bc <- BoxCoxME(Reaction ~ Days + (Days | Subject), data = sleepstudy,
+                       order = 10)
R> logLik(sleep_bc)

'log Lik.' -858 (df=15)
```

Note that the log-likelihood of this model is higher than that of the normal linear model because we are now approximating the baseline transformation function flexibly at the expense of a larger number of parameters.

The conditional quantiles calculated – using the same set of function calls, and with the same caveats, as in the analogous case of LmME – from the model defined by Equation (5) are shown in Figure 4. Comparing these results to Figure 2 reveals departures from conditional normality in the response distributions: At different lengths of sleep deprivation, the conditional distribution of the participants' reaction times is not a shifted normal distribution anymore, but it also changes its spread and shape.

The conditional distributions of the outcome can be further inspected visually with the plot method of **tramME**, which is designed to plot these distributions on a scale specified by the user. The left-hand side plot in Figure 5 compares the conditional densities of subjects 308 and 309 at various sleep deprivation lengths. Clearly, subject 308 is hardly affected by sleep deprivation because the mean and variance of the distribution of reaction time for this subject increase only marginally with days of sleep deprivation. In contrast, subject 309 showed longer mean reaction times and an increased variability of reaction times with increasing duration of sleep deprivation. The variance effect is not detectable from a classical normal linear mixed model but can be observed after a data-driven response transformation to normality. In the right panel of Figure 5, the conditional distribution of a hypothetical reference subject with zero random effects values is depicted. It should be noted that the latter is, in general, not equal to the marginal distribution of the outcome, which can be calculated by integrating the conditional distributions over the vector of random effects. We will return to this question at the end of this section.

The plots in Figure 5 can be generated with the commands

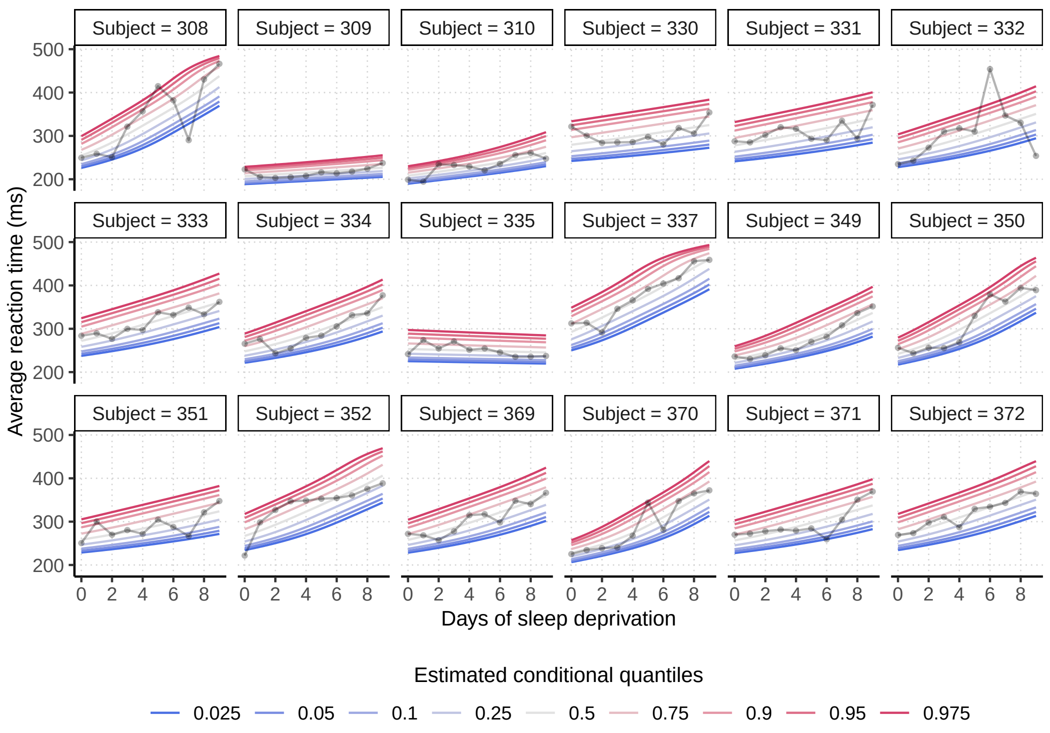


Figure 4: Quantiles of the conditional distribution of the outcome fitted to the `sleepstudy` data with the non-normal (Box-Cox-type) linear mixed-effects transformation model (`BoxCoxME`) defined in Equation (5).

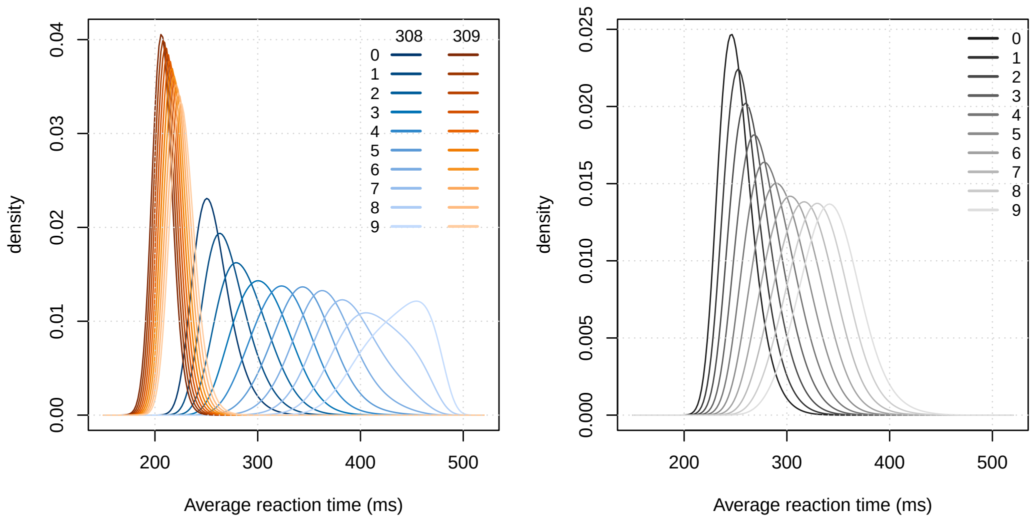


Figure 5: Conditional densities of the outcome from a non-normal (Box-Cox-type) mixed-effects linear transformation model (`BoxCoxME`) fitted to the `sleepstudy` data. *Left:* The conditional densities of subject 308 and 309 at various lengths of sleep deprivation (0-9 days). *Right:* The conditional densities of a reference subject (with random effects equal to zero) at various lengths of sleep deprivation (0-9 days).

```
R> ## -- Compare two subjects (308 and 309)
R> nd <- subset(sleepstudy, subset = Subject %in% c(308, 309))
R> plot(sleep_bc1b, newdata = nd, type = "density", K = 200)
R> ## -- The reference subject (at the mean of the random effect vector)
R> ## (we only need an arbitrary subject)
R> nd <- subset(sleepstudy, subset = Subject == 308)
R> ## NOTE: we explicitly set the random effects vector to 0
R> plot(sleep_bc1b, newdata = nd, ranef = "zero", type = "density", K = 200)
```

(and with some additional formatting steps that are omitted for the sake of brevity but can be found in the accompanying material).

In line with the methodology presented by Hothorn et al. (2018) and Hothorn (2020), we can define more complex mixed-effects transformation models by interacting the covariates with the basis expansion of the outcome. In the resulting extended model, the fixed effects are dependent on the level of the outcome. For the `sleepstudy` example, this model can be written as

$$\mathbb{P}\left(\texttt{Reaction} \leq y \mid \texttt{Days}, \alpha_i, \beta_i\right) = \Phi\left(\mathbf{a}(y)^\top \boldsymbol{\vartheta} - \beta(y)\texttt{Days} - \gamma_{1i} - \gamma_{2i}\texttt{Days}\right) \tag{6}$$
$$\begin{pmatrix}\gamma_{1i}\\ \gamma_{2i}\end{pmatrix} \sim \mathcal{N}_2\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\tau_1^2 & \tau_{12}\\ \tau_{12} & \tau_2^2\end{pmatrix}\right\},$$

which is often referred to as "distribution regression" (Chernozhukov et al., 2013).

The model in Equation 6 can be defined in **tramME** using the | operator on the left-hand side of the model formula.

```
R> sleep_bc2 <- BoxCoxME(Reaction | Days ~ 1 + (Days | Subject), data = sleepstudy,
+                        order = 10, support = c(150, 520))
R> logLik(sleep_bc2)

'log Lik.' -853 (df=25)
```

Plotting the conditional quantiles calculated from the resulting model in Figure 6 and comparing it with Figures 2 and 4 demonstrates the increased flexibility of the specification.

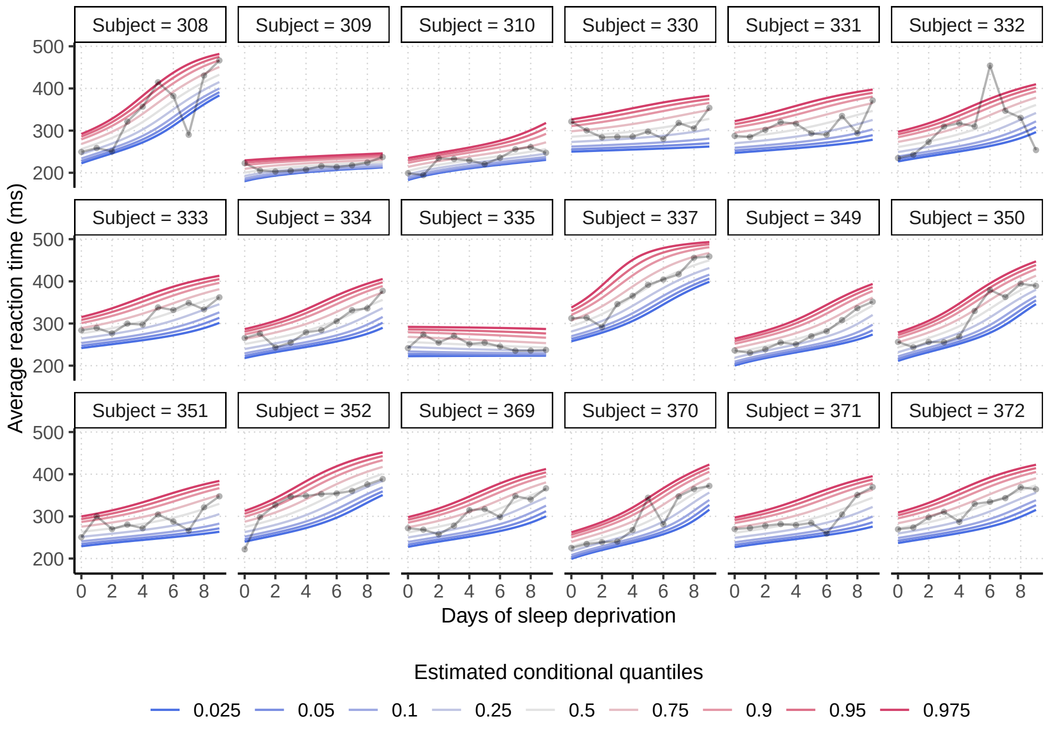


Figure 6: Quantiles of the conditional distribution of the outcome fitted to the `sleepstudy` data with the non-normal (Box-Cox-type) distributional mixed-effects transformation model (`BoxCoxME`) defined in Equation (6).

In many cases, the goal of the analysis is to estimate the *marginal* distribution of the outcomes, i.e., integrating out the random effects from the conditional model (1). In the general formulation, there is no analytical solution for the integral, but we can use numerical methods to approximate the marginal distributions at various values of the outcome. The following code utilizes the `simulate` and `predict` methods implemented in the **tramME** package to get Monte Carlo estimates of the outcome distribution implied by the model (5).

```
R> ndraws <- 1000 ## number of MC draws
R> ## Set up the grid on which we evaluate the marginal distribution
R> nd <- expand.grid(
+    Reaction = seq(min(sleepstudy$Reaction), max(sleepstudy$Reaction),
+                   length.out = 100),
+    Days = 0:9,
+    Subject = 1)
R> ## Sample from the distribution of the random effects
R> re <- simulate(sleep_bc, newdata = nd, nsim = ndraws, what = "ranef", seed = 100)
R> ## Evaluate the conditional distribution at each draw
R> ## (done in parallel to speed up computations)
R> cp <- parallel::mclapply(re, function(x) {
+    predict(sleep_bc, newdata = nd, ranef = x, type = "distribution")
+  }, mc.cores = 8)
R> cp <- array(unlist(cp), dim = c(100, 10, ndraws))
R> ## Integral: take the average over these
R> mp_bc <- apply(cp, c(1, 2), mean)
```

Figure 7 compares the conditional distributions obtained by integrating over the vector of random effects in models (3) and (5).

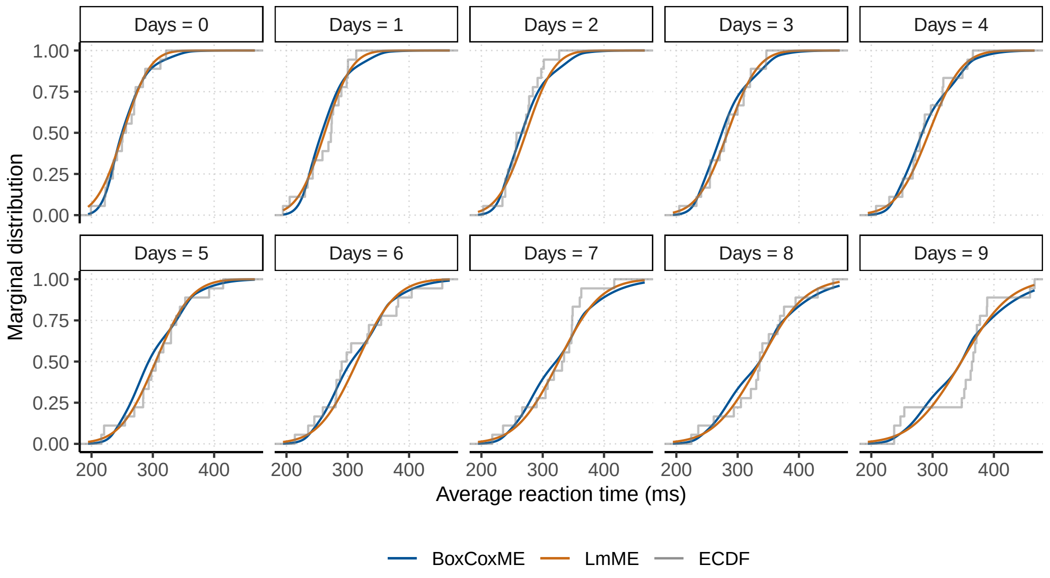


Figure 7: Comparison of the marginal distributions implied by the normal linear (`LmME`) and the Box-Cox-type (`BoxCoxME`) mixed-effects models fitted to the sleep deprivation study dataset. The empirical cumulative distribution functions (ECDF) are also plotted, conditionally on the days of sleep deprivation.

Mixed-effects continuous outcome logistic regression

The increased flexibility of the Box-Cox-type model, i.e., using a general baseline transformation function instead of a linear one, comes with the price that the coefficient estimates will not be easily interpretable as expected changes in the mean in the conditional model. Switching to the standard logistic error distribution provides a solution to this problem, as the parameter estimates in the resulting model can be interpreted as log-odds ratios. This *continuous outcome logistic regression* model was used by Lohse et al. (2017) to analyze body mass index (BMI) distributions.

Manuguerra and Heller (2010) proposed a mixed-effects logistic regression model for bounded, continuous measurements of pain levels in a randomized, double-blind, placebo-controlled trial of low-level laser therapy for subjects with chronic neck pain presented by Chow et al. (2006). The levels of pain, measured on a visual analog scale, and normalized between 0 and 1, are plotted in Figure 8 for each subject at the different follow-up times.

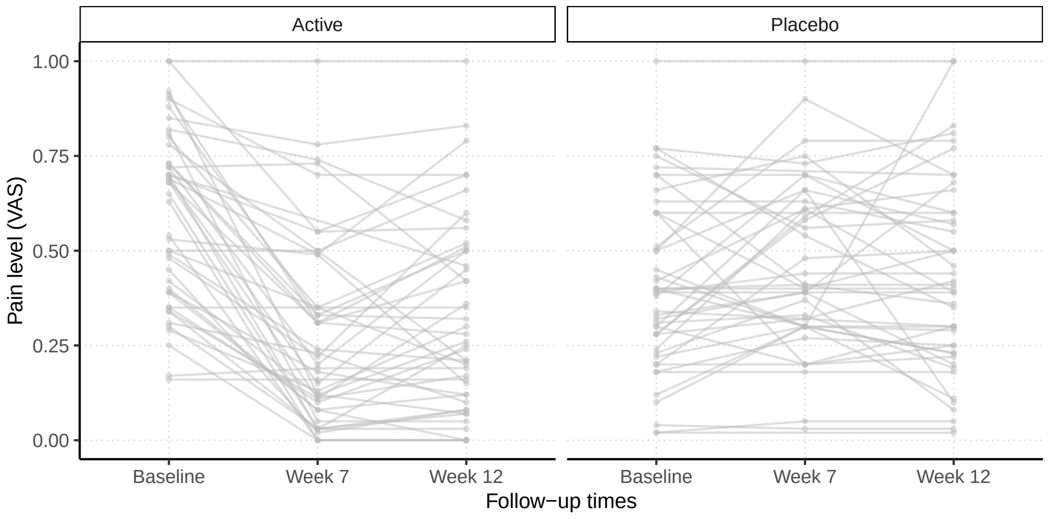


Figure 8: Neck pain dataset: Trajectories of pain levels measured on a visual analog scale (VAS) in the active treatment and placebo-controlled groups reported in Chow et al. (2006).

The mixed-effects model suggested by Manuguerra and Heller (2010) parameterizes the log-odds of experiencing smaller pain levels as a linear function of fixed and random effects and the baseline transformation. With the treatment group indicator, `laser`, and `time` denoting the follow-up times,

$$\log\left[\frac{\mathbb{P}\left(\texttt{pain} \le y \mid \texttt{laser},\texttt{time},\alpha_i\right)}{\mathbb{P}\left(\texttt{pain} > y \mid \texttt{laser},\texttt{time},\alpha_i\right)}\right] = h(y) + \beta_{\text{Active}} + \beta_{7\text{w}} + \beta_{12\text{w}} + \beta_{7\text{w, Active}} + \beta_{12\text{w, Active}} + \alpha_i$$
$$\alpha_i \sim \mathcal{N}(0,\tau^2),$$

where $h(y)$ is an increasing function of the outcome. Rearranging the terms in the model above reveals that this indeed is a mixed-effects transformation model, with the distribution function of the standard logistic distribution ("expit" function) as F_Z,

$$\begin{aligned}\mathbb{P}\left(\texttt{pain} \le y \mid \texttt{laser},\texttt{time},\alpha_i\right) = \operatorname{expit}\big(&h(y) + \beta_{\text{Active}} + \beta_{7\text{w}} + \beta_{12\text{w}} \\ &+ \beta_{7\text{w,Active}} + \beta_{12\text{w, Active}} + \alpha_i\big) \\ \alpha_i \sim \mathcal{N}(0,\tau^2).&\end{aligned} \tag{7}$$

The `ColrME()` function of the **tramME** package estimates mixed-effects continuous outcome logistic regression models using polynomials in Bernstein form to approximate $h(y)$. Applying this model to the `neck_pain` dataset:

```
R> neck_tr <- ColrME(vas ~ laser * time + (1 | id), data = neck_pain,
+                    bounds = c(0, 1), support = c(0, 1))
```

Notice that we explicitly set the bounds and the support of the outcome variable to $[0, 1]$ because the pain levels are measured on a bounded scale.

The **ordinalCont** package by Manuguerra et al. (2020) implements an alternative formulation of the model (7) based on the method described in Manuguerra et al. (2017). In their approach, the baseline transformation is parameterized using B-splines and the estimation is carried out in a penalized likelihood framework.

```
R> library("ordinalCont")
R> neck_ocm <- ocm(vas ~ laser * time + (1 | id), data = neck_pain, scale = c(0, 1))
```

Figure 9 compares the results of the mixed-effects transformation model approach to the estimates obtained using the **ordinalCont** package. Because the two models are not exactly the same, we see some differences in the parameter estimates as well as in the fitted baseline transformation functions, but the two model fits are reasonably close to each other.

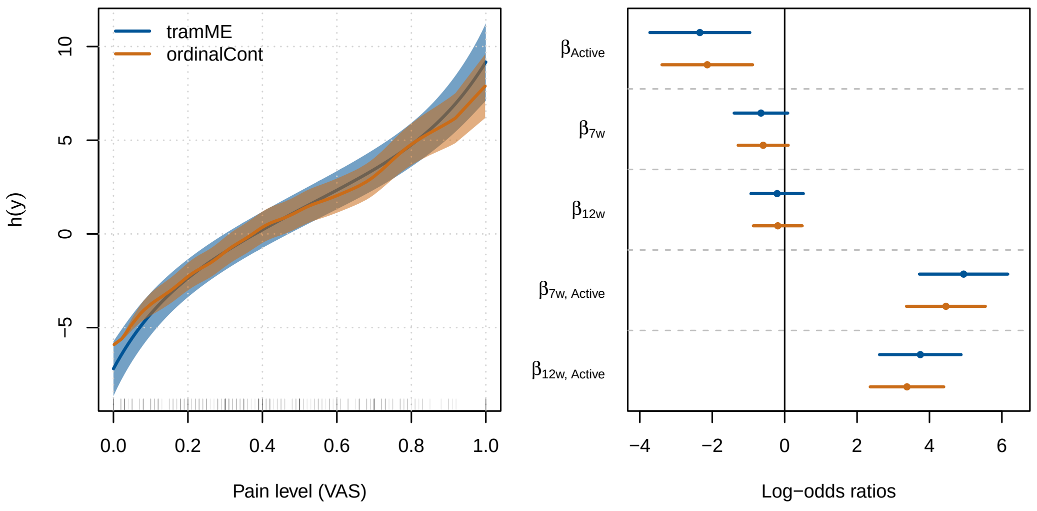


Figure 9: *Left:* Baseline transformations in continuous outcome logistic regressions estimated with the **tramME** and **ordinalCont** packages on the neck_pain dataset. The solid lines denote the point estimates, and the areas indicate the 95% point-wise confidence intervals. *Right:* Coefficient estimates from **tramME** and **ordinalCont** packages and their 95% Wald confidence intervals.

The odds ratio estimates of the model fitted by ColrME(),

```
R> exp(coef(neck_tr))

     laser1       time2       time3 laser1:time2 laser1:time3
     0.0961      0.5200      0.8125     140.2076      42.4076
```

as well as the results from the **ordinalCont** package, suggest that there is an imbalance in the sample at baseline, i.e., the odds of experiencing less pain in the active treatment group is only about 10% that of in the control group for any pain levels. Based on the estimates, the treatment has a strong significant effect, especially at the seven-week follow-up, but seems to level off after 12 weeks.

If we want to compare the marginal distributions in the treatment and control groups directly, we have to average over the distribution of the random effects. Because we only have a random intercept in this example, we have to evaluate a one-dimensional integral. We could use the same Monte Carlo method as we did in Section 2.3.2, or we can apply the adaptive quadrature method implemented in the **stats** package of R. The example below uses this approach to demonstrate the multiple options the user has in dealing with such problems.

```
R> ## A function to evaluate the joint cdf of the response and the random effects:
R> ## Takes a vector of random effect and covariates values, evaluates the conditional
R> ## distribution at these values and multiplies it with the pdf of the random effects
R> jointCDF <- function(re, nd, mod) {
+    nd <- nd[rep(1, length(re)), ]
+    nd$id <- seq(nrow(nd)) ## to take vector-valued REs
+    pr <- predict(mod, newdata = nd, ranef = re, type = "distribution") *
+      dnorm(re, 0, sd = sqrt(varcov(mod)[[1]][1, 1]))
+    c(pr)
+  }
R> ## Marginalize the joint cdf by integrating out the random effects
R> ## using adaptive quadrature
R> marginalCDF <- function(nd, mod) {
+    nd$cdf <- integrate(jointCDF, lower = -Inf, upper = Inf, nd = nd, mod = mod)$value
+    nd
+  }
R> ## Set up the grid on which we evaluate the marginal distribution
R> nd <- expand.grid(vas = seq(0, 1, length.out = 100),
+                    time = unique(neck_pain$time),
+                    laser = unique(neck_pain$laser))
R> ## Calls marginalCDF on each row of nd
R> ## (done in parallel to speed up computations)
R> mp_colr <- parallel::mclapply(split(nd, seq(nrow(nd))),
+                                marginalCDF, mod = neck_tr, mc.cores = 8)
R> mp_colr <- do.call("rbind", mp_colr)
```

Figure 10 compares the marginal distributions at different time points and confirms our previous conclusions on baseline imbalance and treatment effect dynamics.

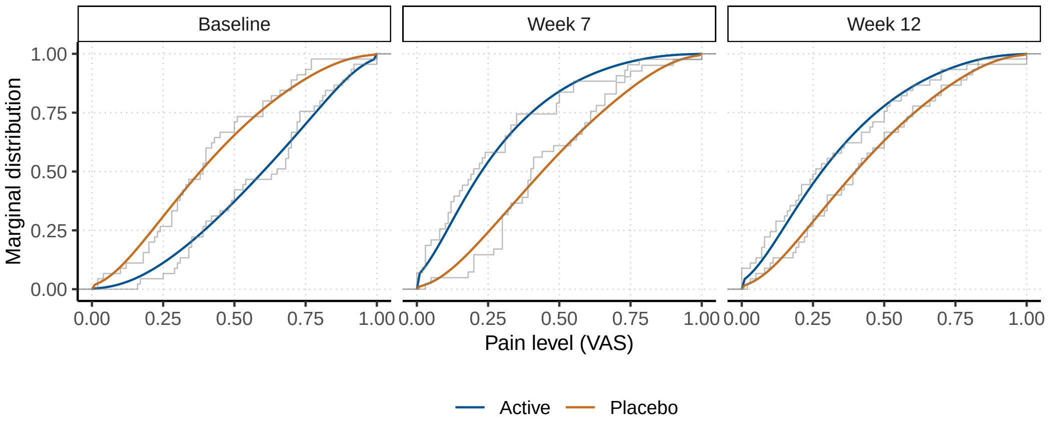


Figure 10: Comparison of marginal distributions, calculated from a mixed-effects continuous outcome logistic regression model in the treatment (Active) and control (Placebo) groups at baseline and the two follow-up times. The step functions represent the empirical cumulative distribution functions of the specific groups.

Mixed-effects transformation models for time-to-event outcomes

Mixed models for right-censored data are important in survival analysis and we consider the example dataset `eortc` in the **coxme** package by Therneau (2020). This simulated dataset emulates the structure of the outcomes of a breast cancer trial by the European Organization for Research and Treatment of Cancer, and consists of 2323, possibly right-censored, data points from 37 enrolling centers. We define a proportional hazards mixed-effects model with random center ($i = 1, \ldots, 37$) and treatment (`trt`) effects (nested within centers and indexed by $j = 0, 1$).

$$\mathbb{P}\left(Y \le y \mid \texttt{trt}, \gamma_{1i}, \gamma_{2j(i)}\right) = 1 - \exp\left(-\exp\left(h(y) + \beta_{\texttt{trt}} + \gamma_{1i} + \gamma_{2j(i)}\right)\right) \tag{8}$$
$$\gamma_{1i} \sim \mathcal{N}(0, \tau_1^2), \quad \gamma_{2j(i)} \sim \mathcal{N}(0, \tau_2^2)$$

This model corresponds to a mixed-effects transformation model with the *minimum extreme value distribution* as the error distribution. Treating the baseline transformation as a general smooth function, approximated using polynomials in Bernstein form, we get the fully parametric version of the Cox proportional hazards model with normally distributed random effects.

We can fit this model with the `CoxphME()` function of **tramME**.

```
R> data("eortc", package = "coxme")
R> eortc$trt <- factor(eortc$trt, levels = c(0, 1))
R> eortc_cp <- CoxphME(Surv(y, uncens) ~ trt + (1 | center/trt), data = eortc,
+                     log_first = TRUE, order = 10)
```

The nested random effects structure is defined with the / operator. The `log_first = TRUE` option casts the outcome variable to the log-scale before defining the Bernstein bases, which usually improves the model fit when dealing with skewed conditional distributions, while we explicitly set the order of the polynomials in Bernstein form with `order = 10`. The confidence interval for the treatment effect (transformed to the hazard ratio scale) suggests evidence for the effectiveness of the treatment,

```
R> exp(confint(eortc_cp, parm = "trt1", estimate = TRUE))
```

```
      lwr  upr  est
trt1 1.8 2.51 2.12
```

while the profile intervals of the random effects standard deviations indicate similar magnitude of center-level and treatment-level (within center) variabilities.

```
R> exp(confint(eortc_cp, pargroup = "ranef", type = "profile", estimate = TRUE,
+              ncpus = 2, parallel = "multicore"))

                           lwr   upr   est
trt:center|(Intercept) 0.0841 0.338 0.208
center|(Intercept)     0.0796 0.384 0.254
```

The transformation model framework by Hothorn (2020) allows for stratification, i.e., specifying separate transformation functions for different groups defined by a stratification factor. Time-dependent effects for the covariates can be introduced in the same way as in distribution regression to relax the proportionality assumption of the Cox model. To check the appropriateness of the proportional hazards assumption between treatment and control groups visually, we re-estimate the model stratifying for the treatment indicator, i.e., fitting transformation functions for the treatment and control groups separately, and inspect whether these two functions, which are the log-cumulative hazards when the error distribution is the minimum extreme value distribution, are parallel.

```
R> eortc_cp2 <- CoxphME(Surv(y, uncens) | 0 + trt ~ 0 + (1 | center/trt), data = eortc,
+                      log_first = TRUE, order = 10)
R> tr <- trafo(eortc_cp2, confidence = "interval")
```

Figure 11 plots the stratified transformation functions against log-time. The two curves are very close to parallel, which indicates that the treatment effect is constant over time, i.e., the proportionality assumption is appropriate in the original model specification.

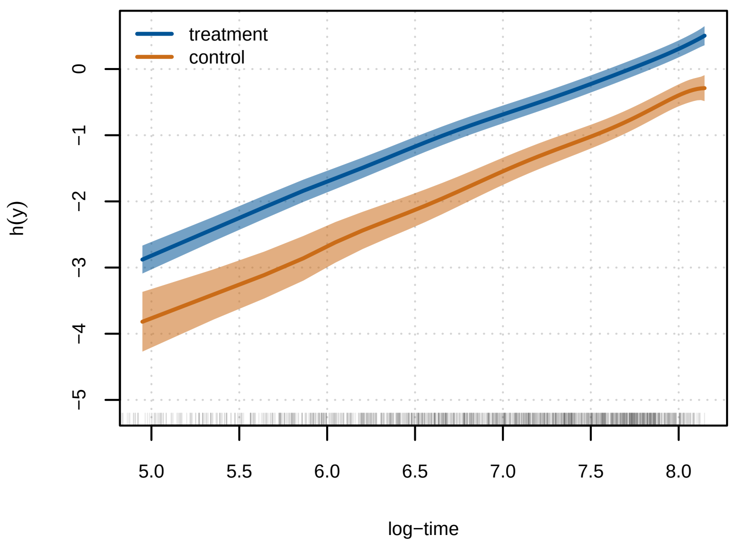


Figure 11: Comparison of baseline transformation functions in treatment and control groups, estimated using a stratified parametric mixed-effects Cox proportional hazards model on the `eortc` dataset.

In addition to proportionality, Figure 11 reveals another important aspect of the data generating process. The fact that the baseline log-cumulative hazards are linear in log-time suggests that the conditional distributions are close to the Weibull distribution, i.e., we can substitute the general baseline transformation function with $h(y) = \vartheta_1 + \vartheta_2 \log(y)$. Flipping the signs of the fixed and random effects terms of (8) and substituting the log-linear function to the baseline transformation, we get the model

$$\mathbb{P}\left(Y \le y \mid \texttt{trt}, \gamma_{1i}, \gamma_{2j(i)}\right) = 1 - \exp\left(-\exp\left(\vartheta_1 + \vartheta_2 \log(y) - \beta_{\texttt{trt}} - \gamma_{1i} - \gamma_{2j(i)}\right)\right)$$
$$\gamma_{1i} \sim \mathcal{N}(0, \tau_1^2), \quad \gamma_{2j(i)} \sim \mathcal{N}(0, \tau_2^2).$$

The `SurvregME()` function of the **tramME** package implements a variety of parametric mixed-effects models that represent specific choices of the error distribution and the baseline transformation function in the general formulation of Equation (1). There are several other R packages available for estimating parametric survival models with mixed effects, such as **parfm** by Munda et al. (2012) and **frailtypack** by Rondeau et al. (2012). However, they typically do not allow for nested random-effects structures when assuming (log-)normally distributed frailty terms.

Fitting a mixed-effects Weibull model to the eortc dataset:

```
R> eortc_w <- SurvregME(Surv(y, uncens) ~ trt + (1 | center/trt), data = eortc,
+                        dist = "weibull")
```

Comparing the parameter estimates of the Cox proportional hazards model to those from the mixed-effects Weibull model,

```
R> ## --- CoxphME
R> c(coef = coef(eortc_cp), se = sqrt(diag(vcov(eortc_cp, pargroup = "shift"))))

coef.trt1   se.trt1
   0.7535    0.0852

R> VarCorr(eortc_cp)


Grouping factor: trt:center (74 levels)
Standard deviation:
(Intercept)
      0.208

Grouping factor: center (37 levels)
Standard deviation:
(Intercept)
      0.254

R> ## --- SurvregME
R> c(coef = -coef(eortc_w), se = sqrt(diag(vcov(eortc_w, pargroup = "shift"))))

coef.trt1   se.trt1
   0.7531    0.0851

R> VarCorr(eortc_w)


Grouping factor: trt:center (74 levels)
Standard deviation:
(Intercept)
      0.208

Grouping factor: center (37 levels)
Standard deviation:
(Intercept)
      0.255
```

as well as their log-likelihood values

```
R> c(logLik(eortc_cp), logLik(eortc_w))

[1] -13027 -13032
```

confirms our suspicion that the dataset was indeed simulated from a conditional Weibull model.

Finally, we can compare the results from **tramME** to parameters estimated with the R package **coxme**.

```
R> library("coxme")
R> eortc_cm <- coxme(Surv(y, uncens) ~ trt + (1 | center/trt), data = eortc)
R> summary(eortc_cm)

Cox mixed-effects model fit by maximum likelihood
  Data: eortc
  events, n = 1463, 2323
  Iterations= 10 54
```

```
                 NULL Integrated Fitted
Log-likelihood -10639     -10518 -10464

                  Chisq   df  p AIC   BIC
Integrated loglik   242  3.0 0 236 220.4
 Penalized loglik   349 39.3 0 270  62.6

Model:  Surv(y, uncens) ~ trt + (1 | center/trt)
Fixed coefficients
      coef exp(coef) se(coef)    z p
trt1 0.742       2.1   0.0827 8.97 0

Random effects
 Group      Variable    Std Dev Variance
 center/trt (Intercept) 0.2045  0.0418
 center     (Intercept) 0.2627  0.0690
```

This package follows a different approach to estimate a mixed-effects Cox model by leaving the baseline hazards unspecified and maximizing the integrated partial likelihood. As a result, the parameter estimates are slightly different from the ones we got using the `CoxphME()` function, but the results are comparable, and the conclusions are identical, nevertheless.

Mixed-effects transformation models for discrete ordinal outcomes

Our last example demonstrates how the mixed-effects transformation framework can be used in modeling correlated discrete ordinal outcomes. As an example, we take the soup tasting dataset by Christensen et al. (2011). The dataset contains 1847 observations from 185 respondents in a soup tasting experiment. The subjects were familiarized with a reference product prior to the experiment and, during the experiment, were asked to distinguish between samples from the reference product and test product using a six-level ordinal scale indicating their level of confidence. The scale ranges from "reference, sure" (`sureness` $= 1$) to "not reference, sure" (`sureness` $= 6$). Figure 12 presents the proportions of response categories for the test and reference samples for respondent groups defined by how often they consume soup.

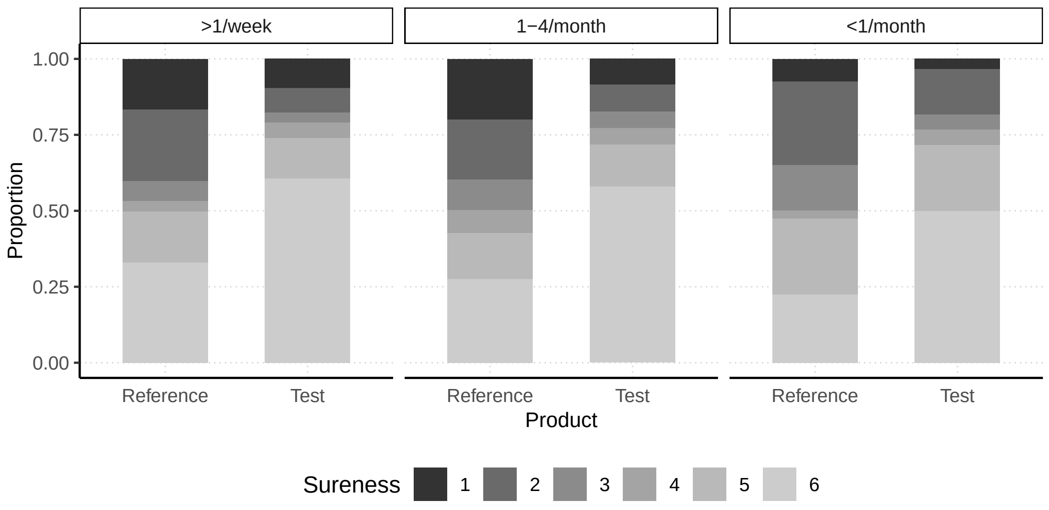


Figure 12: Data from the soup tasting study reported by Christensen et al. (2011): Proportions of responses of sureness levels (six levels, ranging from "reference, sure" to "not reference, sure") after tasting test and reference products. The data points are grouped by how often the respondents consume soup (more than once a week, one to four times a month, less than once a month).

Let us assume that we are interested in comparing the distributions of sureness ratings for reference products and test products while taking the repeated nature of the design into account. Moreover, in doing so, we also want to control for how often the respondents usually consume soup (denoted by the covariate `freq`). With $k = 1 \ldots 5$, indicating the sureness levels except the last one, $i = 1 \ldots, 185$ indexing the respondents, and $j = 0, 1$ indexing the reference and test products (covariate `prod`), respectively, the regression model we estimate can be written as

$$\begin{aligned} \mathbb{P}\left(\texttt{sureness} \le k \mid \texttt{prod}, \texttt{freq}, \gamma_{1i}, \gamma_{2j(i)}\right) &= \Phi\big(\vartheta_k - \beta_{\text{test}} - \beta_{\text{1-4/month}} - \beta_{\text{<1/month}} \\ &\qquad - \gamma_{1i} - \gamma_{2j(i)}\big) \\ \gamma_{1i} \sim \mathcal{N}(0, \tau_1^2), &\quad \gamma_{2j(i)} \sim \mathcal{N}(0, \tau_2^2). \end{aligned} \tag{9}$$

The `PolrME()` function of the **tramME** package estimates models for ordered discrete outcomes. Depending on the choice of the error distribution, the user can fit proportional odds (logistic distribution), ordinal probit (standard normal distribution), proportional hazards (minimum extreme value distribution), or cumulative maximum extreme value models. In our example, we set `method = 'probit'` to estimate the probit model,

```
R> soup_pr <- PolrME(SURENESS ~ PROD + SOUPFREQ + (1 | RESP/PROD),
+                    data = soup, method = "probit")
R> logLik(soup_pr)

'log Lik.' -2666 (df=10)
```

The R package **ordinal** by Christensen (2019) also implements mixed-effects regression models for ordered discrete outcomes. As a cross-check, we can re-estimate the same model with the function `clmm()`,

```
R> library("ordinal")
R> soup_or <- clmm(SURENESS ~ PROD + SOUPFREQ + (1 | RESP/PROD), data = soup,
+                  link = "probit")
R> logLik(soup_or)

'log Lik.' -2666 (df=10)
```

Based on the likelihood values and the parameter estimates,

```
R> max(abs(coef(soup_or) - coef(soup_pr, with_baseline = TRUE)))

[1] 1.76e-05
```

the results are essentially the same.

We can introduce non-proportional effects in the transformation model framework by stratifying on a covariate. In our example, we might want to extend the model to allow for different effect sizes of the soup consumption frequency covariate, depending on the level of the outcome variable. Rewriting model (9),

$$\begin{aligned} \mathbb{P}\left(\texttt{sureness} \le k \mid \texttt{prod}, \texttt{freq}, \gamma_{1i}, \gamma_{2j(i)}\right) &= \Phi\left(\vartheta_k - \beta_{\text{test}} - \beta_{\text{1-4/month},k} - \beta_{\text{<1/month},k} - \gamma_{1i} - \gamma_{2j(i)}\right) \\ \gamma_{1i} \sim \mathcal{N}(0, \tau_1^2), &\quad \gamma_{2j(i)} \sim \mathcal{N}(0, \tau_2^2), \end{aligned}$$

and estimating it with **tramME** by stratifying for the soup frequency factor

```
R> soup_pr2 <- PolrME(SURENESS | SOUPFREQ ~ PROD + (1 | RESP/PROD),
+                     data = soup, method = "probit")
R> logLik(soup_pr2)

'log Lik.' -2655 (df=18)
```

The likelihood ratio test comparing the two specifications suggests some evidence that the extended, partially proportional model fits the data better.

```
R> anova(soup_pr, soup_pr2)

Model comparison

         Model 1: SURENESS ~ PROD + SOUPFREQ + (1 | RESP/PROD)
         Model 2: SURENESS | SOUPFREQ ~ PROD + (1 | RESP/PROD)

        npar logLik  AIC  BIC Chisq Chisq df Pr(>Chisq)
Model 1   10  -2666 5352 5408
Model 2   18  -2655 5347 5446  21.9        8     0.0051 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Discussion

Building the implementation of mixed-effects transformation models on the package **TMB** leads to significant efficiency gains in the computationally intensive steps of the maximum likelihood estimation. This computational efficiency is partly due to the use of Laplace approximation to integrate over the vector of random effects. However, several sources point out that Laplace's method can lead to biased estimates in some distributional settings. Pinheiro and Chao (2006) provide detailed numerical comparisons of the Laplacian approximation to adaptive Gaussian quadrature algorithms in the context of multilevel generalized models. Joe (2008) evaluates the method in the case of discrete outcome mixed-effects models and concludes that the inaccuracy increases with the amount of discreteness of the response variable and decreases as the cluster sizes increase.

It is worth mentioning that the conditional approach of modeling the distribution of the response, which is the basis of the transformation models implemented in the **tramME** package, is not the only way one could approach the problem of correlated outcomes in regression settings. The main alternative to a conditional (mixed-effects) modeling approach is a marginal model that parameterizes the marginal distribution of the outcome and treats the covariance structure as nuisance parameters. Generalized estimating equations (GEE, Hardin and Hilbe, 2013) models represent prominent examples of such an approach. Proponents of marginal models point out that, in a conditional model, the fixed effects parameter estimates cannot be interpreted as population averages, which is usually of primary interest in a regression analysis. Lindsey and Lambert (1998) emphasize that marginal parameter estimates from longitudinal studies can only be interpreted as population averages when the participants are representative to their populations, which is usually not the case. Moreover, they argue that defining models based on marginal distributions very often leads to complicated and implausible conditional distributions, whereas conditional models can more easily express physiologically plausible mechanisms on the level of the individual. Lee and Nelder (2004) argue that conditional models are more fundamental as they allow for both marginal and conditional inferences, which is not true in the case of marginal models. As we demonstrated in Sections 2.3.2 and 2.3.3, the marginal distributions implied by the conditional transformation model can be easily approximated using numerical techniques.

The **tramME** package, introduced in this article, extends the available options for modeling grouped data structures with mixed-effects regressions in several ways: Through its dense code base, **tramME** provides a unified and efficient estimation framework for a broad range of regression models. Examples in Section 2.3 demonstrate that using this single package, several very specific regression problems can be addressed. Relying only on a limited number of packages, in turn, decreases the likelihood of errors in the statistical analysis. As the examples show, the modular structure of our approach naturally leads to extensions of existing models (such as accounting for censoring or introducing nested or crossed random effects structures) that would otherwise require a lot of effort to re-implement from scratch. Moreover, the underlying theory of linear transformation models provides a flexible basis for the implementation of the package and for its future extensions.

Acknowledgments

This work was supported by the Swiss National Science Foundation, grant number 200021_184603.

Bibliography

A. Almohaimeed and J. Einbeck. *boxcoxmix: Box-Cox-Type Transformations for Linear and Logistic Models with Random Effects*, 2020. URL https://CRAN.R-project.org/package=boxcoxmix. R package version 0.28. [p405]

D. Bates, M. Mächler, B. M. Bolker, and S. Walker. Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. URL https://doi.org/10.18637/jss.v067.i01. [p398]

G. Belenky, N. J. Wesensten, D. R. Thorne, M. L. Thomas, H. C. Sing, D. P. Redmond, M. B. Russo, and T. J. Balkin. Patterns of performance degradation and restoration during sleep restriction and subsequent recovery: a sleep dose-response study. *Journal of Sleep Research*, 12(1):1–12, 2003. URL https://doi.org/10.1046/j.1365-2869.2003.00337.x. [p400, 401]

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964. URL https://doi.org/10.1111/j.2517-6161.1964.tb00553.x. [p405]

M. E. Brooks, K. Kristensen, K. J. van Benthem, A. Magnusson, C. W. Berg, A. Nielsen, H. J. Skaug, M. Mächler, and B. M. Bolker. glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, 9(2):378–400, 2017. URL https://doi.org/10.32614/RJ-2017-066. [p400]

V. Chernozhukov, I. Fernández-Val, and B. Melly. Inference on counterfactual distributions. *Econometrica*, 81(6):2205–2268, 2013. URL https://doi.org/10.3982/ECTA10582. [p407]

R. T. Chow, G. Z. Heller, and L. Barnsley. The effect of 300 mW, 830 nm laser on chronic neck pain: a double-blind, randomized, placebo-controlled study. *Pain*, 124(1-2):201–210, 2006. URL https://doi.org/10.1016/j.pain.2006.05.018. [p409]

R. H. B. Christensen. *ordinal—Regression Models for Ordinal Data*, 2019. URL https://CRAN.R-project.org/package=ordinal. R package version 2019.12-10. [p415]

R. H. B. Christensen, G. Cleaver, and P. B. Brockhoff. Statistical and Thurstonian models for the A-not A protocol with and without sureness. *Food Quality and Preference*, 22(6):542–549, 2011. URL https://doi.org/10.1016/j.foodqual.2011.03.003. [p414]

E. Demidenko. *Mixed models: theory and applications with R*. Wiley series in probability and statistics. Wiley, second edition, 2013. URL https://doi.org/10.1002/9781118651537. [p398]

J. W. Hardin and J. M. Hilbe. *Generalized estimating equations*. CRC Press, second edition, 2013. URL https://doi.org/10.1201/b13880. [p416]

T. Hothorn. Most likely transformations: The mlt package. *Journal of Statistical Software*, 92(1):1–68, 2020. URL https://doi.org/10.18637/jss.v092.i01. [p398, 407, 412]

T. Hothorn and L. Barbanti. *tram: Transformation Models*, 2021. URL https://CRAN.R-project.org/package=tram. R package version 0.6-0. [p398]

T. Hothorn, L. Möst, and P. Bühlmann. Most likely transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018. URL https://doi.org/10.1111/sjos.12291. [p398, 400, 405, 407]

H. Joe. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics & Data Analysis*, 52(12):5066–5074, 2008. URL https://doi.org/10.1016/j.csda.2008.05.002. [p416]

K. Kristensen, A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell. TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, 70(5):1–21, 2016. URL https://doi.org/10.18637/jss.v070.i05. [p400]

Y. Lee and J. A. Nelder. Conditional and Marginal Models: Another View. *Statistical Science*, 19(2): 219–238, 2004. URL https://doi.org/10.1214/088342304000000305. [p416]

J. K. Lindsey and P. Lambert. On the appropriateness of marginal models for repeated measurements in clinical trials. *Statistics in Medicine*, 17(4):447–469, 1998. URL https://doi.org/10.1002/(SICI)1097-0258(19980228)17:4<447::AID-SIM752>3.0.CO;2-G. [p416]

T. Lohse, S. Rohrmann, D. Faeh, and T. Hothorn. Continuous outcome logistic regression for analyzing body mass index distributions. *F1000Research*, 6:1933, 2017. URL https://doi.org/10.12688/f1000research.12934.1. [p408]

M. Manuguerra and G. Heller. Ordinal regression models for continuous scales. *The International Journal of Biostatistics*, 6:14–14, 2010. URL https://doi.org/10.2202/1557-4679.1230. [p408, 409]

M. Manuguerra, G. Heller, and J. Ma. Semi-parametric ordinal regression models for continuous scales. In M. Grzegorczyk and G. Ceoldo, editors, *Proceedings of the 32nd International Workshop on Statistical Modelling*, volume 1, pages 236–241, 2017. [p409]

M. Manuguerra, G. Z. Heller, and J. Ma. Continuous ordinal regression for analysis of visual analogue scales: The R package ordinalCont. *Journal of Statistical Software*, 96(8):1–24, 2020. URL https://doi.org/10.18637/jss.v096.i08. [p409]

M. Munda, F. Rotolo, and C. Legrand. parfm: Parametric frailty models in R. *Journal of Statistical Software*, 51(11):1–20, 2012. URL https://doi.org/10.18637/jss.v051.i11. [p413]

J. Pinheiro and E. C. Chao. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15(1): 58–81, 2006. URL https://doi.org/10.1198/106186006X96962. [p416]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2021. URL `https://CRAN.R-project.org/package=nlme`. R package version 3.1-152. [p398]

V. Rondeau, Y. Mazroui, and J. R. Gonzalez. frailtypack: An R package for the analysis of correlated survival data with frailty models using penalized likelihood estimation or parametrical estimation. *Journal of Statistical Software*, 47(4):1–28, 2012. URL `https://doi.org/10.18637/jss.v047.i04`. [p413]

S. G. Self and K.-Y. Liang. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association*, 82(398): 605–610, 1987. URL `https://doi.org/10.1080/01621459.1987.10478472`. [p405]

B. Tamási and T. Hothorn. *tramME: Transformation Models with Mixed Effects*, 2021. URL `https://CRAN.R-project.org/package=tramME`. R package version 0.1.2. [p398]

T. M. Therneau. *coxme: Mixed Effects Cox Models*, 2020. URL `https://CRAN.R-project.org/package=coxme`. R package version 2.2-16. [p411]

T. M. Therneau. *A Package for Survival Analysis in R*, 2021. URL `https://CRAN.R-project.org/package=survival`. R package version 3.2-12. [p403]

J. T. Thorson and K. Kristensen. Implementing a generic method for bias correction in statistical models using random effects, with spatial and population dynamics examples. *Fisheries Research*, 175:66–74, 2016. URL `https://doi.org/10.1016/j.fishres.2015.11.016`. [p402]

Y. Tian, T. Hothorn, C. Li, F. E. Harrell Jr., and B. E. Shepherd. An empirical comparison of two novel transformation models. *Statistics in Medicine*, 39(5):562–576, 2020. URL `https://doi.org/10.1002/sim.8425`. [p398]

Bálint Tamási and Torsten Hothorn
Institut für Epidemiologie, Biostatistik und Prävention
Universität Zürich
Hirschengraben 84, CH-8001 Zürich
Switzerland
ORCiD: 0000-0002-2629-7362, 0000-0001-8301-0471
`balint.tamasi@uzh.ch`, `Torsten.Hothorn@R-project.org`

miRecSurv Package: Prentice-Williams-Peterson Models with Multiple Imputation of Unknown Number of Previous Episodes

by David Moriña, Gilma Hernández-Herrera and Albert Navarro

Abstract Left censoring can occur with relative frequency when analyzing recurrent events in epidemiological studies, especially observational ones. Concretely, the inclusion of individuals that were already at risk before the effective initiation in a cohort study may cause the unawareness of prior episodes that have already been experienced, and this will easily lead to biased and inefficient estimates. The **miRecSurv** package is based on the use of models with specific baseline hazard, with multiple imputation of the number of prior episodes when unknown by means of the COMPoisson distribution, a very flexible count distribution that can handle over, sub, and equidispersion, with a stratified model depending on whether the individual had or had not previously been at risk, and the use of a frailty term. The usage of the package is illustrated by means of a real data example based on an occupational cohort study and a simulation study.

Introduction

It is not unusual in cohort epidemiological studies that part (or all) of the participants have experienced the event under study at least once before the beginning of the follow-up. This situation is particularly common in the case of observational designs. Under these circumstances, the prior history and prior time at risk of these individuals can be unknown or estimated on the basis of self recall questionnaires, which could lead to modeling issues when the baseline hazard of suffering the event is time-dependent. If the event of interest can only occur once, and it occurred for an individual before the start of the follow-up, the result for this individual is fixed regardless of the duration of the follow-up. Therefore we are in the well-known and well-studied situation of left censoring of a binary variable, for which specific modeling techniques are available. On the other hand, if the event of interest can be suffered several times by the same individual (it is recurrent), and the number of events suffered by the individuals in the cohort before the beginning of the follow-up is unknown, we face a left censoring situation with a discrete censored variable that can define different baseline hazards depending on the episode an individual is at risk of.

This paper introduces the **miRecSurv**, useful when the prior history is unknown for all, or some, of the individuals included in a cohort and the outcome of interest is a recurrent event with event dependence. Specifically, we suppose that we know the moment from where all individuals are at risk, but the number of episodes experienced by the individuals in this time is unknown. This is a realistic situation in practice; for example, it is very probable that in a workforce cohort, we know when a worker started to work (thus, to be at risk of having a sick leave) and, however, and especially for people with ample trajectory, that we do not know whether or not in effect they have already had sick leaves (and in this case, how many). Another situation that might deal with this issue is a study of cohorts with an outcome of incidence of infection from human papillomavirus on adult women. It would be relatively simple to know how long they have been at risk (beginning of active sexual life) or to make a reasonable assumption. However, we will not be able to know the number of infections since most of the time, when they occur, they are asymptomatic (Fernández-Fontelo et al., 2016).

Methods

Theoretical approach

Our proposal starts from the assumption that even though the previous history of all or some of the individuals is unknown, we do know which of these were at risk prior to the beginning of the follow-up and starting when. Further, that is based fundamentally on three considerations: 1) impute k, the number of previous episodes for those subjects at risk before the beginning of the follow-up; 2) treat the subpopulation of subjects "Previously at risk" separately from those "Not previously at risk", and 3) use a frailty term basically to capture the error that will be made when imputing k. Concretely, in the two formulations, "Counting process" (Eq. 1) and "Gap time" (Eq. 2), the ones we call "Specific Hazard Frailty Model Imputed", in its versions - Counting Process (SHFMI.CP) and Gap Time (SHFMI.GT):

$$\lambda_{ikr}(t) = \nu_i \lambda_{0kr}(t) e^{X_i \beta} \tag{1}$$

$$\lambda_{ikr}(t) = \nu_i \lambda_{0kr}(t - t_{k-1}) e^{X_i \beta}, \tag{2}$$

where k will be the number of previous episodes of individual i in case they are known or their imputed value in case they are not known; r indicates the subpopulation the individual belongs to: "Previously at risk" or "Not previously at risk". In both cases, information corresponding to the time to risk prior to $t = 0$ for each individual is included in $X_i\beta$, which will be zero for all those that start to be at risk as of the beginning of the follow-up and a value different than zero for those that were previously at risk. In practice, this proposal means that we stratify by the interaction between having been at risk or not before the beginning of the follow-up and the number of previous episodes.

Therefore, the use of the term individual random error ν_i intends to capture the error that will be made when imputing and the effect of any variable that having a potential effect, would not have been considered in the analysis. Stratifying by the number of prior episodes intends to safeguard the event dependence. Doing it as an interaction with the fact that it is an individual previously at risk, or not, separates the two subpopulations to not mix times that are not comparable, on the same scale. More details on the proposed methodology can be found in Hernández-Herrera et al. (2021).

The imputation of the number of previous events in individuals at risk before the beginning of the follow-up is done through the COMPoisson generalized distribution (using the log link function), that allows adjusting a regression model using the Conway-Maxwell Poisson (COMPoisson) distribution (Shmueli et al., 2005) considering the dispersion of the data (sub, equi, or overdispersion). This imputation is carried out through multiple imputation calculating its parameters directly from the observed data in two phases: Firstly, a generalized linear model (GLM) is fitted using the number of episodes observed during the follow-up as the response variable and the covariates selected by the user as explanatory, based on the COMPoisson distribution. Imputed values are randomly sampled from this distribution with the parameters obtained in the previous step, including random noise generated from a normal distribution. In order to produce a proper estimation of uncertainty, the described methodology is included in a multiple imputation framework, according to the well known Rubin's rules (Rubin, 1987) and based on the following steps in a Bayesian context:

1. Fit the COMPoisson count data model and find the posterior mean and variance $\hat{\beta}$ and $V(\hat{\beta})$ of model parameters β.
2. Draw new parameters β^* from $N(\hat{\beta}, V(\hat{\beta}))$.
3. Compute predicted scores p using the parameters obtained in the previous step.
4. Draw imputations from the COMPoisson distribution and scores obtained in the previous step.

The COMPoisson random number generation is based on the `rcom` function included in the archived package **compoisson** (Dunn, 2012). The overperformance of the COMPoisson distribution in this context compared to alternative discrete distributions (Poisson, negative binomial, Hermite and zero-inflated versions) is discussed in Hernández-Herrera et al. (2020) by means of a comprehensive simulation study.

The miRecSurv package

The main function of the **miRecSurv** is `recEvFit`, which allows the user to fit recurrent events survival models. A call to this function might be

```
recEvFit(formula, data, id, prevEp, riskBef, oldInd,
         frailty=FALSE, m=5, seed=NA, ...)
```

The description of these arguments can be summarized as follows:

- `formula`: a formula object, with the response on the left of a $\sim$ operator and the terms on the right. The response must be a survival object as returned by the `Surv` function.
- `data`: a data.frame in which to interpret the variables named in the formula.
- `id`: subject identifier.
- `prevEp`: known previous episodes.
- `riskBef`: indicator for new individual in the cohort (`riskBef=FALSE`) or subject who was at risk before the start of follow-up (`riskBef=TRUE`).

- `oldInd`: time an individual has been at risk prior to the follow-up. This time can be positive or negative (time origin as the start of follow-up).
- `frailty`: should the model include a frailty term. Defaults to `FALSE`.
- `m`: number of multiple imputations. The default is `m=5`.
- `seed`: an integer that is used as argument by the `set.seed` function for offsetting the random number generator. Default is to leave the random number generator alone.
- `...`: extra arguments to pass to `coxph`.

The output of this function is a list with seven elements:

- `fit`: a list with all the `coxph` objects fitted for each imputed dataset.
- `coeff`: a list with the vectors of coefficients from the models fitted to each imputed dataset.
- `loglik`: a list with the loglikelihood for each model fitted.
- `vcov`: a list with the variance-covariance matrices for the parameters fitted for each of the imputed datasets.
- `AIC`: a list with the AIC of each of the models fitted.
- `CMP`: summary tables of the fitted COMPoisson models used for imputing missing values.
- `data.impute`: the original dataset with the multiple imputed variables as final columns.

In order to facilitate the interpretation, a pooled summary table is available via `summary` method, formatted in a very similar way to the summary tables of very well-known functions as `coxph`, as can be seen in the next section. As in some cases the multiple imputation process might be computationally expensive and take some time, the function `recEvFit` provides the user with a progress bar.

Examples

Simulation study

To illustrate our proposal, we use simulated data generated with the parameters estimated in a worker cohort, where the outcome is the occurrence of sick leave due to any cause. Table 1 shows the characteristics of each episode in this population, estimated in a cohort study described in Navarro et al. (2012). The maximum number of episodes that a subject may suffer was not fixed, although the baseline hazard was considered constant when $k \geq 4$. X_1, X_2, and X_3 are covariates that represent the exposure, with $X_i \sim Bernoulli(0.5)$, $i = 1, 2, 3$, and $\beta_1 = 0.25$, $\beta_2 = 0.5$, and $\beta_3 = 0.75$ being their parameters that represent effects of different magnitudes, set independently of the episode k to which the worker is exposed. All the simulations were conducted using the R package **survsim** (Moriña and Navarro, 2014), and all the code to reproduce these analyses is available as supplementary material.

Episode	Distribution	β_0	Ancillary
1	Log-logistic	7.974	0.836
2	Weibull	7.109	0.758
3	Log-normal	5.853	1.989
4	Log-normal	5.495	2.204

Table 1: Characteristics of the simulated population.

To illustrate the usage of the **miRecSurv** package, the results corresponding to a sample from the first scenario can be obtained by means of

```
library(survsim)
library(miRecSurv)
d.ev    <- c('llogistic','weibull','weibull','weibull')
b0.ev   <- c(5.843, 5.944, 5.782, 5.469)
a.ev    <- c(0.700, 0.797, 0.822, 0.858)
d.cens  <- c('weibull','weibull','weibull','weibull')
b0.cens <- c(7.398, 7.061, 6.947, 6.657)
a.cens  <- c(1.178, 1.246, 1.207, 1.422)
```

```
set.seed(1234)
sample1 <- rec.ev.sim(n=1500, foltime=1095,
                      dist.ev=d.ev, anc.ev=a.ev, beta0.ev=b0.ev,
                      dist.cens=d.cens, anc.cens=a.cens, beta0.cens=b0.cens,
                      beta=list(c(-.25,-.25,-.25,-.25), c(-.5,-.5,-.5,-.5), c(-.75,-.75,-.75,-.75)),
                      x=list(c("bern", .5), c("bern", .5), c("bern", .5)),
                      priskb=.1, max.old=5475)
sample1$old2 <- -sample1$old
sample1$old2[is.na(sample1$old)] <- 0

### Shared frailty
ag_s1 <- coxph(Surv(start2,stop2,status)~as.factor(x)+as.factor(x.1)+as.factor(x.2)+old2+
               strata(as.factor(risk.bef))+frailty(nid), data=sample1)

### Counting process
shfmi.cp_s1 <- recEvFit(Surv(start2, stop2, status)~x+x.1+x.2, data=sample1,
                        id="nid", prevEp = "obs.episode",
                        riskBef = "risk.bef", oldInd = "old", frailty=TRUE, m=5, seed=1234)

### Gap time
shfmi.gt_s1 <- recEvFit(Surv(stop2-start2, status)~x+x.1+x.2, data=sample1,
                        id="nid", prevEp = "obs.episode",
                        riskBef = "risk.bef", oldInd = "old", frailty=TRUE, m=5, seed=1234)
```

The generated cohort, including the estimated number of previous events (multiple imputed), can be obtained as

```
head(shfmi.cp_s1$data.impute)
  nid real.episode obs.episode       time status    start     stop      time2   start2
1   1            1           1 119.673502      1   0.0000 119.6735 119.673502 124.5052
2   1            2           2 389.637244      1 119.6735 509.3107 389.637244 244.1787
3   1            3           3 244.130315      1 509.3107 753.4411 244.130315 633.8160
4   1            4           4 144.476145      0 753.4411 897.9172 144.476145 877.9463
5   2            1           1 107.943850      1   0.0000 107.9438 107.943850 681.4178
6   2            2           2   6.472291      1 107.9438 114.4161   6.472291 789.3617
      stop2 old risk.bef long z x x.1 x.2 EprevCOMPoissDef1 EprevCOMPoissDef2
1  244.1787   0    FALSE   NA 1 1   0   0                 0                 0
2  633.8160   0    FALSE   NA 1 1   0   0                 1                 1
3  877.9463   0    FALSE   NA 1 1   0   0                 2                 2
4 1022.4224   0    FALSE   NA 1 1   0   0                 3                 3
5  789.3617   0    FALSE   NA 1 1   1   0                 0                 0
6  795.8340   0    FALSE   NA 1 1   1   0                 1                 1
  EprevCOMPoissDef3 EprevCOMPoissDef4 EprevCOMPoissDef5
1                 0                 0                 0
2                 1                 1                 1
3                 2                 2                 2
4                 3                 3                 3
5                 0                 0                 0
6                 1                 1                 1
```

The pooled coefficients table can be obtained as

```
summary(shfmi.cp_s1)
Call:
recEvFit(formula = Surv(start2, stop2, status) ~ x + x.1 + x.2,
    data = sample1, id = "nid", prevEp = "obs.episode", riskBef = "risk.bef",
    oldInd = "old", frailty = TRUE, m = 5, seed = 1234)

Coefficients:
                 coef exp(coef)  se(coef)        Chisq     Pr(>|z|)
x        2.766059e-01 1.3186465 0.3161915  76.49190798 1.401017e-17
x.1      4.337341e-01 1.5430085 0.3010033 174.86750236 2.970463e-39
x.2      7.489812e-01 2.1148444 0.5144217 436.60796033 8.875830e-95
old     -2.584687e-05 0.9999742 0.4976880   0.03005029 7.830972e-01
frailty            NA        NA        NA  77.64460457 6.481409e-02
```

```
AIC:  34867.6

COM-Poisson regression model for imputing missing values
              Estimate     SE   z.value  Pr(>|z|)
(Intercept)    -6.3611 0.0348 -182.5394 0.000e+00
x               0.2356 0.0240    9.8376 7.755e-23
x.1             0.3563 0.0258   13.7869 3.054e-43
x.2             0.6446 0.0305   21.1139 5.927e-99
S:(Intercept)  -0.5029 0.0315  -15.9637 2.288e-57
```

Simulated data include four scenarios of $n = 1500$ subjects, with a maximum follow-up time of 3 years and a maximum time at risk prior to the beginning of the cohort of 15 years. The first scenario has a 10% of subjects at risk prior to the beginning of the cohort (i.e., we do not know the number of previous episodes in 10% of the subjects), whilst the second, third, and fourth samples have 25%, 50%, and 100%, respectively. Samples based on these settings were generated 100 times and the estimates were averaged across simulations for each sample.

To compare the results obtained, we also estimate the shared frailty model (Eq 3). This model has a common hazard baseline (i.e., does not consider the event dependence) and incorporates an individual frailty term.

$$\lambda_{ikr}(t) = \nu_i \lambda_0(t) e^{X_i \beta} \tag{3}$$

Results of fitting the described models in scenario 1 are summarized in the Table 2:

Model	$\hat{\beta_1}$	$SE(\hat{\beta_1})$	$\hat{\beta_2}$	$SE(\hat{\beta_2})$	$\hat{\beta_3}$	$SE(\hat{\beta_3})$
Shared Frailty	0.292	0.043	0.558	0.043	0.843	0.043
SHFMI.CP	0.244	0.034	0.470	0.036	0.706	0.039
SHFMI.GT	0.218	0.030	0.419	0.031	0.632	0.034

Table 2: Average estimates obtained on scenario 1 (10% of subjects at risk prior to the beginning of the cohort).

Below are presented the results for the second scenario, Table 3:

Model	$\hat{\beta_1}$	$SE(\hat{\beta_1})$	$\hat{\beta_2}$	$SE(\hat{\beta_2})$	$\hat{\beta_3}$	$SE(\hat{\beta_3})$
Shared Frailty	0.287	0.039	0.554	0.039	0.832	0.040
SHFMI.CP	0.242	0.032	0.472	0.035	0.703	0.040
SHFMI.GT	0.217	0.028	0.425	0.030	0.633	0.034

Table 3: Average estimates obtained on scenario 2 (25% of subjects at risk prior to the beginning of the cohort).

Results for scenario 3, Table 4:

Model	$\hat{\beta_1}$	$SE(\hat{\beta_1})$	$\hat{\beta_2}$	$SE(\hat{\beta_2})$	$\hat{\beta_3}$	$SE(\hat{\beta_3})$
Shared Frailty	0.277	0.033	0.542	0.034	0.818	0.035
SHFMI.CP	0.243	0.030	0.474	0.036	0.710	0.042
SHFMI.GT	0.217	0.025	0.421	0.029	0.632	0.033

Table 4: Average estimates obtained on scenario 3 (50% of subjects at risk prior to the beginning of the cohort).

Finally, Table 5 shows the estimates when 100% of the subjects are at risk prior to the beginning of the follow-up:

The average relative bias of the estimates produced by each method is shown in Figure 1. It can be seen that the estimates produced by the **miRecSurv** are less biased than those based on the common baseline hazard model.

Model	$\hat{\beta}_1$	$SE(\hat{\beta}_1)$	$\hat{\beta}_2$	$SE(\hat{\beta}_2)$	$\hat{\beta}_3$	$SE(\hat{\beta}_3)$
Shared Frailty	0.262	0.026	0.527	0.027	0.797	0.029
SHFMI.CP	0.248	0.029	0.495	0.040	0.742	0.053
SHFMI.GT	0.208	0.022	0.416	0.027	0.626	0.034

Table 5: Average estimates obtained on scenario 4 (100% of subjects at risk prior to the beginning of the cohort).

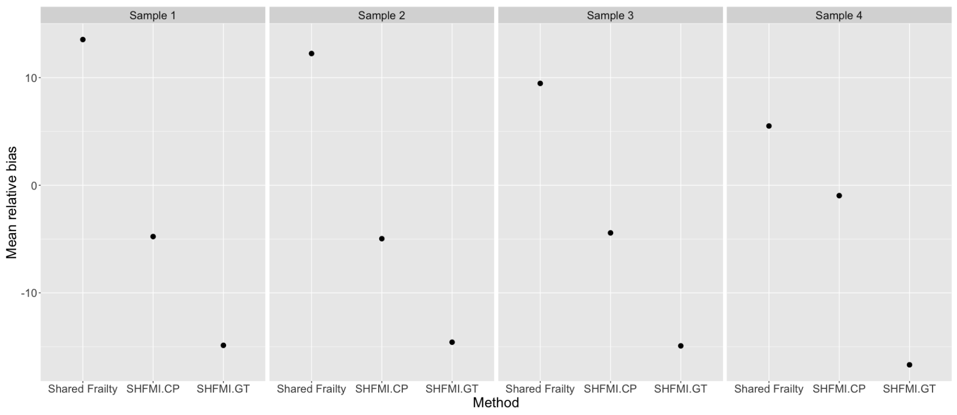


Figure 1: Average relative bias of the estimates produced by method (Shared Frailty, SHFMI.CP, SHFMI.GT) in each scenario (extreme left: 10% of subjects at risk prior to the beginning of the cohort; left: 25%; right: 50%; extreme right: 100%).

Example

The real example corresponds to a selected sample of the HC-UFMG cohort (Reis et al., 2008), which involves workers of a public hospital in Belo Horizonte, Brazil. Specifically, we selected all workers present on January 1st, 2004 ($n = 512$) and all of them whose contract with the hospital was signed from that date until the end of the follow-up, December 31st, 2007 ($n = 884$). The outcome of interest was the occurrence of sick leave for any diagnosis, with three covariates: *contract* (civil servant; non civil servant), *type of work* (healthcare professionals; non-healthcare) and *sex* (female; male). The incidence rate was 11.7 sick leaves per 100 worker-months, and the median of follow-up per worker was 24.3 months. Figure 2 represents a random subsample of 30 workers from this cohort, and it can be seen that some of the workers were at risk of suffering the event of interest before the start of the follow-up, and these episodes cannot be seen and must be estimated. Time 0 is the start of the follow-up of each worker, being January 1st, 2004 for all those who were contracted in the hospital before that date, or the starting date of their contract among those who were contracted later.

To analyze the association of sex, type of contract, and the kind of job developed by these workers over the risk of suffering sick leaves, the package **miRecSurv** can be used in the following way.

```
mod1 <- recEvFit(Surv(start, stop, status)~Male+Non.civil.servant+
                                           Non.healthcare, data=example,
                 id="nid", prevEp="num", riskBef="prev2", oldInd="days_prev",
                 frailty=TRUE, m=5, seed=1234)
summary(mod1)
Call:
recEvFit(formula = Surv(start, stop, status) ~ Male + Non.civil.servant +
    Non.healthcare, data = example, id = "nid", prevEp = "num",
    riskBef = "prev2", oldInd = "days_prev", frailty = TRUE,
    m = 5, seed = 1234)

Coefficients:
                           coef exp(coef)  se(coef)      Chisq     Pr(>|z|)
Male              -0.2966733104 0.7432868 0.2033092   24.386339 3.999350e-07
Non.civil.servant -0.1788549829 0.8362272 0.1320313    9.221709 2.391622e-03
Non.healthcare    -0.1315412853 0.8767431 0.1318741    5.568327 1.703076e-02
days_prev         -0.0002383671 0.9997617 0.2027526    6.457974 7.089736e-03
```

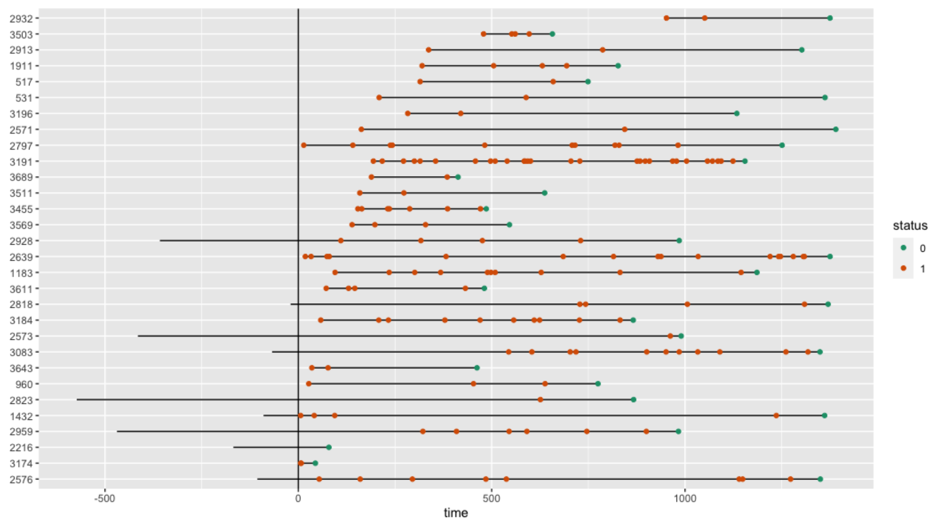


Figure 2: Evolution in time of a subsample of 30 workers from the HC-UFMG cohort. The vertical line represents the start of follow-up.

```
frailty                       NA         NA         NA 1157.231797 5.664063e-35

AIC:  29699

COM-Poisson regression model for imputing missing values
                  Estimate     SE   z.value  Pr(>|z|)
(Intercept)        -6.7602 0.0407 -166.0266 0.000e+00
Male               -0.1665 0.0261   -6.3766 1.810e-10
Non.civil.servant  -0.0132 0.0246   -0.5354 5.924e-01
Non.healthcare     -0.0917 0.0246   -3.7297 1.917e-04
S:(Intercept)      -1.4043 0.0709  -19.8059 2.646e-87
```

According to these results, it can be seen that all three considered explanatory variables are significantly associated with the risk of suffering sick leaves (see Coefficients table), although the type of contract is not significant on the model used to impute the number of episodes suffered before the start of the follow-up.

Conclusion

Left censoring, when analyzing recurrent events, is a situation that can occur with a certain frequency in cohort studies. For example, it will occur in every cohort that follows subjects that were already at risk of experiencing the outcome of interest before the beginning of the follow-up. If the recurrent event presents event dependence, not knowing the number of episodes that each individual has had prior to the effective initiation of the follow-up, and analyzing this through "classical" methods, is an important problem, as it will lead to biased and inefficient estimates.

The presented proposal seems to work reasonably well, outperforming the alternative to a common hazard model. It is important to carry out a comprehensive study to evaluate the performance of the presented proposal. It is also worth noticing that although in this particular example, the average relative biases produced by the shared frailty model are relatively low, this is not always the case, as can be seen in a simulation study like Navarro et al. (2017).

Unavailability of the number of previous episodes that an individual has had should not be a justification for the use of models with a common baseline hazard which on the large majority of occasions show a higher bias and less coverage than specific baseline hazard models, as shown by the results of this work.

Bibliography

J. Dunn. *compoisson: Conway-Maxwell-Poisson Distribution*, 2012. URL `https://CRAN.R-project.org/package=compoisson`. R package version 0.3. [p420]

A. Fernández-Fontelo, A. Cabaña, P. Puig, and D. Moriña. Under-reported data analysis with INAR-hidden Markov chains. *Statistics in Medicine*, 35(26):4875–4890, nov 2016. ISSN 10970258. doi: 10.1002/sim.7026. [p419]

G. Hernández-Herrera, A. Navarro, and D. Moriña. Regression-based imputation of explanatory discrete missing data. *arXiv preprint*, 2020. URL http://arxiv.org/abs/2007.15031. [p420]

G. Hernández-Herrera, D. Moriña, and A. Navarro. Left-censored recurrent event analysis in epidemiological studies: a proposal when the number of previous episodes is unknown. *arXiv preprint*, 2021. URL http://arxiv.org/abs/2102.11279. [p420]

D. Moriña and A. Navarro. The R package survsim for the simulation of simple and complex survival data. *Journal of Statistical Software*, 59(2):1–20, 2014. URL http://www.jstatsoft.org/v59/i02/. [p421]

A. Navarro, D. Moriña, R. Reis, F. B. Nedel, M. Martín, and S. Alvarado. Hazard functions to describe patterns of new and recurrent sick leave episodes for different diagnoses. *Scandinavian journal of work, environment & health*, 38(5):447–55, sep 2012. ISSN 1795-990X. doi: 10.5271/sjweh.3276. URL http://www.ncbi.nlm.nih.gov/pubmed/22286954. [p421]

A. Navarro, G. Casanovas, S. Alvarado, and D. Moriña. Analyzing recurrent events when the history of previous episodes is unknown or not taken into account: proceed with caution. *Gaceta sanitaria*, 31(3):227–234, 2017. [p425]

R. J. Reis, P. d. F. La Rocca, L. Basile, A. Navarro, and M. Martín. Cohort profile: The hospital das clínicas cohort study, belo horizonte, minas gerais, brazil. *International Journal of Epidemiology*, 37(4): 227–234, 2008. [p424]

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley, 1987. ISBN 047108705X. [p420]

G. Shmueli, T. P. Minka, J. B. Kadane, S. Borle, and P. Boatwright. A useful distribution for fitting discrete data: revival of the Conway–Maxwell–Poisson distribution. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(1):127–142, jan 2005. ISSN 1467-9876. doi: 10.1111/J.1467-9876.2005.00474.X. URL https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.1467-9876.2005.00474.x. [p420]

David Moriña
Department of Econometrics, Statistics and Applied Economics - Riskcenter (IREA), Universitat de Barcelona (UB)
Avinguda Diagonal, 690 (08034) Barcelona
Spain
Centre de Recerca Matemàtica (CRM)
ORCID: 0000-0001-5949-7443
dmorina@ub.edu

Gilma Hernández-Herrera
Instituto de Investigaciones Médicas, Facultad de Medicina, Universidad de Antioquia
Carrera 51 D No. 62-29. Edificio Manuel Uribe Ángel, Medellín
Colombia
PhD program in Methodology of Biomedical Research and Public Health. Autnomous University of Barcelona (UAB)
Avinguda de Can Domènech, S/N (08193) Cerdanyola del Vallès
Spain
gilma.hernandez@udea.edu.co

Albert Navarro
Research group on Psychosocial Risks, Organization of Work and Health (POWAH), Autonomous University of Barcelona (UAB)
Biostatistics Unit, Faculty of Medicine, Autonomous University of Barcelona (UAB)
Avinguda de Can Domènech, S/N (08193) Cerdanyola del Vallès
Spain
albert.navarro@uab.cat

An R package for Non-Normal Multivariate Distributions: Simulation and Probability Calculations from Multivariate Lomax (Pareto Type II) and Other Related Distributions

by Zhixin Lun and Ravindra Khattree

Abstract Convenient and easy-to-use programs are readily available in R to simulate data from and probability calculations for several common multivariate distributions such as normal and t. However, functions for doing so from other less common multivariate distributions, especially those which are asymmetric, are not as readily available, either in R or otherwise. We introduce the R package **NonNorMvtDist** to generate random numbers from multivariate Lomax distribution, which constitutes a very flexible family of skewed multivariate distributions. Further, by applying certain useful properties of multivariate Lomax distribution, multivariate cases of generalized Lomax, Mardia's Pareto of Type I, Logistic, Burr, Cook-Johnson's uniform, F, and inverted beta can be also considered, and random numbers from these distributions can be generated. Methods for the probability and the equicoordinate quantile calculations for all these distributions are then provided. This work substantially enriches the existing R toolbox for nonnormal or nonsymmetric multivariate probability distributions.

Introduction

A k-dimensional multivariate Lomax (Pareto Type II) probability distribution was first introduced by (Nayak, 1987) as a joint distribution of k skewed nonnegative random variables $X_1, \cdots, X_k$ with joint probability density function given by

$$f(x_1,\ldots,x_k) = \frac{\left[\prod_{i=1}^{k}\theta_i\right] a(a+1)\cdots(a+k-1)}{\left(1+\sum_{i=1}^{k}\theta_i x_i\right)^{a+k}}, \quad x_i > 0, a, \theta_i > 0, i = 1,\ldots,k. \tag{1}$$

We will denote above density function by $\mathrm{ML}_k(a;\theta_1,\ldots,\theta_k)$. Prior to Nayak (1987), the bivariate case of multivariate Lomax distribution was studied by Lindley and Singpurwalla (1986). Nayak (1987) indicated that the k-dimensional multivariate Lomax distribution could be obtained by mixing k independent univariate exponential distributions with different failure rates with the mixing parameter η that has a gamma distribution with certain shape parameter a and the scale parameter 1. This fact readily provides an approach to simulate the multidimensional random vectors from the multivariate Lomax distribution. The multivariate Lomax distribution is also transformable to many other useful multivariate distributions, and therefore, simulations from these distributions are also easily accomplished. Similarly, with appropriate transformations or reparameterizations (or otherwise directly from the probability density function (pdf)), we can also accomplish the cumulative probability calculations as well as the calculation of equicoordinate quantiles. The objective of this work is to formalize all of the above and to provide a ready-to-use R package titled **NonNorMvtDist** for practitioners to efficiently execute the same. See Lun and Khattree (2020).

The objective of our work is to enrich the existing R packages for supporting simulation and computations for nonnormal continuous multivariate distributions. To the best of our knowledge and also from the list of packages for multivariate distributions in CRAN (`https://cran.r-project.org/web/views/Distributions.html`), there is no package that provides both simulation and probability computations for multivariate Pareto distribution. In our package, we provide functions for doing so to both Lomax (Pareto type II) and Mardia's Pareto type I distributions. For multivariate logistic distribution, package **VGAM** (Yee, 2019) implements the bivariate logistic distribution while we support p-variate logistic distribution for $p > 2$. Moreover, multivariate Burr, F, and inverted beta distributions had not been implemented in R until we included them in the package **NonNorMvtDist**.

This paper is organized as follows. In Section Multivariate Lomax and related distributions, we provide simulation algorithms for generating data from k-dimensional multivariate Lomax and generalized Lomax (to be defined later) distributions. Through transformations, the tasks of random numbers generation from Mardia's multivariate Pareto Type I, Logistic, Burr, Cook-Johnson's uniform, and F are achieved. Based on the remarks from Balakrishnan and Lai (2009), we then extend Nayak's work to simulate data from the multivariate inverted beta distribution. In Section Probability computations, we discuss numerical computations of cumulative distribution functions of equicoordinate quantiles and survival functions for the above distributions. In Section Illustrations of simulations and probability calculations, we illustrate the use of respective functions as implemented in R for each of the above distributions for the bivariate ($k = 2$) case. In Section Computation times, we provide a run-time study to assess the computation times for functions as the dimension of data increases. In Section Maximum likelihood estimation of parameters, we implement maximum likelihood estimation of parameters for these distributions. In Section Two applications, we give two applications of package **NonNorMvtDist**, namely, (i) data generation from certain nonelliptical symmetric multivariate distributions with univariate normal marginals and (ii) computation of critical values of the multivariate F distribution. Section Concluding remarks includes some concluding remarks, pointing out other applications.

Multivariate Lomax and related distributions

Multivariate Lomax distribution can be derived as the probability distribution of a k-component system where k independent exponential random variables have a common environment or mixing parameter following a gamma distribution with shape parameter a and scale parameter b. Let the corresponding random vector be $\mathbf{X} = (X_1, \cdots, X_k)'$. The probability density function of $\mathbf{X}$ is given by (1), and the joint survival function of $\mathbf{X}$ is

$$S(x_1, \ldots, x_k) = \left(1 + \sum_{i=1}^{k} \theta_i x_i\right)^{-a}, \quad x_i > 0, a > 0, \theta_i > 0, i = 1, \ldots, k. \tag{2}$$

Specifically, the pivotal result that we use is given by the following theorem (see Nayak (1987)),

Theorem 2.1: Conditioned on fixed mixing parameter η, representing the environment effect, let $X_1, \cdots, X_k$ be independent exponentially distributed random variables with failure rates $\eta\lambda_1, \cdots, \eta\lambda_k$, respectively. Let the environment effect η be distributed as a Gamma random variate with probability density

$$g(\eta) = b^a \exp(-\eta b)\eta^{a-1}/\Gamma(a), \eta > 0, a, b > 0.$$

Then, the unconditional joint density of $X_1, \cdots, X_k$ is given by (1), where $\theta_i = \lambda_i/b, i = 1, \cdots, k$. Clearly, without loss of generality, b can be taken as 1, in which case $\theta_i = \lambda_i, i = 1, \cdots, k$.

In view of the above result, we implement the simulation from k-dimensional multivariate Lomax distribution by adopting the following algorithm.

Algorithm-ML$_k(a; \theta_1, \ldots, \theta_k)$**:**

1. Generate a random number η from Gamma$(a, 1)$ distribution;
2. With η as generated in Step 1, generate k-independent random variables X_i, $i = 1, \ldots, k$, each from exponential distribution with parameter $\eta\theta_i$, $i = 1, \ldots, k$, respectively. Let $\mathbf{X} = (X_1, X_2, \cdots, X_k)'$;
3. To obtain a random sample of size n, repeat the Steps 1 and 2 n times.

Nayak (1987) also generalized this distribution by mixing conditionally independent X_i having the Gamma$(l_i, \eta\theta_i)$ distribution, with mixing variable $\eta \sim$ Gamma$(a, 1)$, $i = 1, \ldots, k$. This is termed as generalized multivariate Lomax distribution denoted by GML$(a; \theta_1, \ldots, \theta_k, l_1, \ldots, l_k)$ and has the probability density function

$$f(x_1, \cdots, x_k) = \frac{\left[\prod_{i=1}^{k} \theta_i^{l_i}\right] \Gamma\left(\sum_{i=1}^{k} l_i + a\right) \prod_{i=1}^{k} x_i^{l_i - 1}}{\Gamma(a) \left[\prod_{i=1}^{k} \Gamma(l_i)\right] \left(1 + \sum_{i=1}^{k} \theta_i x_i\right)^{\sum_{i=1}^{k} l_i + a}}, \quad x_i > 0, a, \theta_i, l_i > 0, i = 1, \cdots, k \tag{3}$$

Accordingly, we perform the corresponding simulation by implementing the suitable changes in the above algorithm. The algorithm is given below.

Algorithm-GML$(a; \theta_1, \ldots, \theta_k, l_1, \ldots, l_k)$**:**

1. Generate a random number η from Gamma$(a, 1)$ distribution;

2. With η as generated in Step 1, generate k-independent X_i, each following Gamma$(l_i, \eta\theta_i)$, $i = 1, \ldots, k$, respectively;
3. To obtain a random sample of size n, repeat the Steps 1 and 2 n times.

Both algorithms are easily implemented using R **stats** (R Core Team, 2019) functions `rexp()` and `rgamma()`, respectively, for generating univariate exponential and gamma random variates. In the following, we describe approaches to generate other distributions related to multivariate Lomax and generalized multivariate Lomax distributions.

Nayak (1987) has also discussed the inter-relationships between many other multivariate distributions and generalized multivariate Lomax distribution. In view of these inter-relationships, the above algorithm can accordingly be amended to simulate data from these distributions - a task which can be quite difficult to accomplish directly. These inter-relationships are described in Table 1. For convenience, we assume $\mathbf{X} = (X_1, \cdots, X_k)' \sim \text{ML}_k(a; \theta_1, \cdots, \theta_k)$ and $\mathbf{T} = (T_1, \cdots, T_k)' \sim \text{GML}_k(a; \theta_1, \cdots, \theta_k; l_1, \cdots, l_k)$.

Multivariate Distribution	**Transformation/ Parameter Substitutions**	**Probability Density Function**
Lomax	$l_i = 1$, $i = 1, \cdots, k$	$f(x_1, \ldots, x_k) = \frac{\left[\prod_{i=1}^k \theta_i\right] a(a+1)\cdots(a+k-1)}{\left(1+\sum_{i=1}^k \theta_i x_i\right)^{a+k}}$ $x_i > 0, \quad a > 0, \quad \theta_i > 0$
Mardia's Pareto Type I	$l_i = 1$, $Y_i = X_i + 1/\theta_i$, $i = 1, \cdots, k$	$f(y_1, \cdots, y_k) = \frac{\left[\prod_{i=1}^k \theta_i\right] a(a+1)\cdots(a+k-1)}{\left(\sum_{i=1}^k \theta_i y_i - k + 1\right)^{a+k}}$, $y_i > 1/\theta_i > 0, \quad a > 0, \quad \theta_i > 0$
Logistic	$l_i = 1$ and $a = 1$, $W_i = \mu_i - \sigma_i \ln(\theta_i X_i)$, $i = 1, \cdots, k$	$f(w_1, \cdots, w_k) = \frac{k! \exp\left(-\sum_{i=1}^k \frac{w_i - \mu_i}{\sigma_i}\right)}{\prod_{i=1}^k \sigma_i \left(1+\sum_{i=1}^k \exp\left(-\frac{w_i-\mu_i}{\sigma_i}\right)\right)^{1+k}}$, $-\infty < w_i, \mu_i < \infty, \quad \sigma_i > 0$
Burr	$l_i = 1$, $B_i = (\theta_i X_i / d_i)^{1/c_i}$, $i = 1, \cdots, k$	$f(b_1, \cdots, b_k) = \frac{\left[\prod_{i=1}^k c_i d_i\right] a(a+1)\cdots(a+k-1)\left[\prod_{i=1}^k b_i^{c_i - 1}\right]}{\left(1+\sum_{i=1}^k d_i b_i^{c_i}\right)^{a+k}}$, $b_i > 0, \quad a > 0, \quad c_i > 0, \quad d_i > 0$
Cook-Johnson's uniform	$l_i = 1$, $V_i = (1 + \theta_i X_i)^{-a}$, $i = 1, \cdots, k$	$f(v_1, \cdots, v_k) = \frac{\Gamma(a+k)}{\Gamma(a) a^k} \prod_{i=1}^k v_i^{(-1/a)-1} \left[\sum_{i=1}^k v_i^{-1/a} - k + 1\right]^{-(a+k)}$, $0 < v_i \leq 1, \quad a > 0$
F with degrees of freedom $(2a, 2l_1, \ldots, 2l_k)$	$\theta_i = l_i / a$, $i = 1, \cdots, k$	$f(t_1, \cdots, t_k) = \frac{\left[\prod_{i=1}^k (l_i/a)^{l_i}\right] \Gamma\left(\sum_{i=1}^k l_i + a\right) \prod_{i=1}^k t_i^{l_i - 1}}{\Gamma(a)\left[\prod_{i=1}^k \Gamma(l_i)\right]\left(1+\sum_{i=1}^k \frac{l_i}{a} t_i\right)^{\sum_{i=1}^k l_i + a}}$, $t_i > 0, \quad a > 0, \quad l_i > 0$
Inverted Beta	$\theta_i = 1$, $i = 1, \cdots, k$	$f(t_1, \cdots, t_k) = \frac{\Gamma\left(\sum_{i=1}^k l_i + a\right) \prod_{i=1}^k t_i^{l_i - 1}}{\Gamma(a)\left[\prod_{i=1}^k \Gamma(l_i)\right]\left(1+\sum_{i=1}^k t_i\right)^{\sum_{i=1}^k l_i + a}}$, $t_i > 0, \quad a > 0, \quad l_i > 0$

Table 1: Multivariate distributions related to $\text{GML}_k(a; \theta_1, \cdots, \theta_k; l_1, \cdots, l_k)$.

The multivariate F distribution can also be obtained by considering

$$T_i = \frac{S_i/(2l_i)}{S_0/(2a)}, \quad i = 1, \ldots, k, \tag{4}$$

where $S_0, S_1, \ldots, S_k$ are independent Chi-square variables with $2a, 2l_1, \ldots, 2l_k$ degrees of freedom respectively; see Johnson and Kotz (1972). It is the joint distribution of the ratios of mean squares under certain linear hypotheses on treatments as discussed in Krishnaiah (1965) in the context of simultaneous ANOVA and MANOVA tests where S_0 is a residual sum of squares and S_i's are various effect sums of squares. The density given in Table 1 is a special case of generalized multivariate F distribution defined by Krishnaiah (1965) when S_i's, $i = 1, \cdots, k$, are all independent. This fact is useful in that the Tables given by Armitage and Krishnaiah (1964) make use of this in constructing the statistical tables for certain linear hypotheses. We use these tables to confirm our calculation as done by our R programs. The multivariate Inverted Beta distribution also called the multivariate inverted Dirichlet distribution, is essentially a special case of multivariate F distribution when $l_1 = l_2 = \cdots = l_k = a$.

Figure 1 pictorially summarizes the relationships between (generalized) Lomax and other distributions.

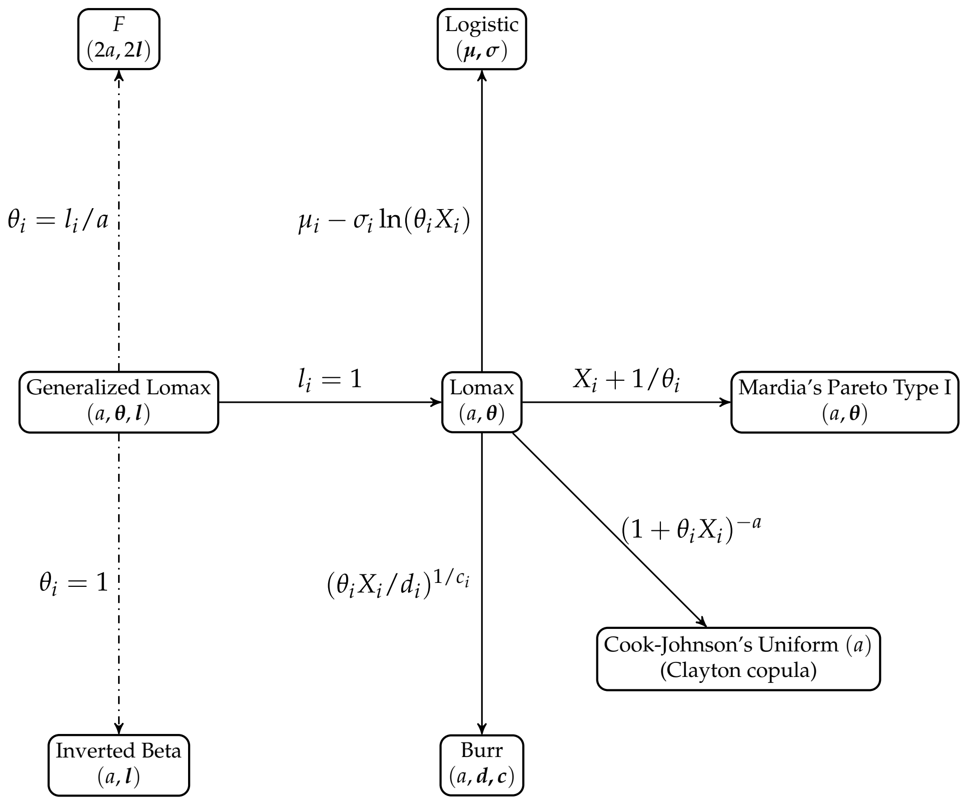

F $(2a, 2\mathbf{l})$
Logistic $(\boldsymbol{\mu}, \boldsymbol{\sigma})$
$\theta_i = l_i/a$
$\mu_i - \sigma_i \ln(\theta_i X_i)$
Generalized Lomax $(a, \boldsymbol{\theta}, \mathbf{l})$
$l_i = 1$
Lomax $(a, \boldsymbol{\theta})$
$X_i + 1/\theta_i$
Mardia's Pareto Type I $(a, \boldsymbol{\theta})$
$(1 + \theta_i X_i)^{-a}$
$\theta_i = 1$
$(\theta_i X_i / d_i)^{1/c_i}$
Cook-Johnson's Uniform (a) (Clayton copula)
Inverted Beta $(a, \mathbf{l})$
Burr $(a, \mathbf{d}, \mathbf{c})$

Figure 1: Relationships among (generalized) multivariate Lomax and other distributions. Solid lines represent transformations and dashed dot lines represent reparameterization (parameter substitutions).

Probability computations

Here, we give details of computations of cumulative probability distribution function (cdf), survival function, equicoordinate quantile function for each distribution introduced in Section Multivariate Lomax and related distributions. Depending on the situation, the calculation may sometimes be simpler for joint cdf or joint survival function.

Distributions transformable from Lomax distribution

The multivariate Lomax distribution has an explicit closed-form expression for the joint survival function given by (2). The survival or cumulative distribution function of other related distributions can be obtained either directly or through appropriate transformations. We summarize these explicit expressions of cumulative distribution function $F(\cdot)$ and survival function $S(\cdot)$ in Table 2.

For the cumulative distribution functions or survival functions with no closed-form expressions, we rely on the following useful formulas (Joe, 1997):

$$S(\mathbf{x}) = 1 + \sum_{C \in \mathcal{C}} (-1)^{|C|} F_C(x_j, j \in C), \tag{5}$$

$$F(\mathbf{x}) = 1 + \sum_{C \in \mathcal{C}} (-1)^{|C|} S_C(x_j, j \in C), \tag{6}$$

where $F_C(x_j, j \in C)$ ($S_C(x_j, j \in C)$) is the joint cdf (joint survival function) of x_j where the subscripts belong to the set C, which is a subset of $\{1, 2, \cdots, k\}$. Clearly, $C \in \mathcal{C}$ where $\mathcal{C}$ is the powerset of $\{1, 2, \cdots, k\}$. Also, $|C|$ represents the cardinality of C.

The equation

$$P[x_i \le q_i, i = 1, \cdots, k] = \int_0^{q_1} \cdots \int_0^{q_k} f(x_1, \cdots, x_k) dx_k \cdots dx_1 = p$$

Multivariate Distribution	Cumulative Distribution Function	Survival Function
Lomax	No closed form	$S(x_1,\cdots,x_k)=\left(1+\sum_{i=1}^k\theta_i x_i\right)^{-a}$ $x_i>0,\quad a,\theta_i>0$
Mardia's Pareto Type I	No closed form	$S(y_1,\cdots,y_k)=\left(\sum_{i=1}^k\theta_i y_i-k+1\right)^{-a}$ $y_i>0,\quad a,\theta_i>0$
Logistic	$F(w_1,\cdots,w_k)=\left[1+\sum_{i=1}^k\exp(-\frac{w_i-\mu_i}{\sigma_i})\right]^{-1}$ $-\infty<w_i,\mu_i<\infty,\quad\sigma_i>0$	No closed form
Burr	No closed form	$S(b_1,\cdots,b_k)=\left(1+\sum_{i=1}^k d_i b_i^{c_i}\right)^{-a}$ $b_i>0,\quad a,d_i,c_i>0$
Cook-Johnson's Uniform	$F(v_1,\cdots,v_k)=\left[\sum_{i=1}^k v_i^{-1/a}-k+1\right]^{-a}$ $0<v_i\le 1,\quad a>0$	No closed form

Table 2: Cumulative distribution functions and survival functions of multivariate Lomax and related distributions.

for a given p does not have a unique solution $(q_1,\cdots,q_k)$. We thus provide the quantile computations only for the equicoordinate quantile, obtained by solving the following equation for q,

$$P[X_i\le q, i=1,\cdots,k]=\int_0^q\cdots\int_0^q f(x_1,\cdots,x_k)dx_k\cdots dx_1=p, \tag{7}$$

where $0<p<1$ is a (given) cumulative probability. We make use of the R **stats** function `uniroot()`, which is used for finding one dimensional root.

Distributions related to generalized multivariate Lomax distribution

For generalized multivariate Lomax distribution and its related distributions, explicit expressions of the cumulative distribution function and survival function are not available. Thus, we obtain the cumulative probabilities through multiple integral in (8) below over the unit cube $[0,1]^k$ by using the adaptive multivariate integration function `hcubature()` in package **cubature** (Narasimhan et al., 2018).

$$\begin{aligned}F(x_1,\ldots,x_k)&=\int_0^{x_1}\cdots\int_0^{x_k}f(t_1,\cdots,t_k)dt_k\cdots dt_1,\\&=\prod_{i=1}^k x_i\int_0^1\cdots\int_0^1 f(u_1x_1,\cdots,u_kx_k)du_k\cdots du_1,\quad(u_i=t_i/x_i, i=1,\cdots,k).\end{aligned} \tag{8}$$

The following result is used for the computation of the cumulative distribution function for the generalized Lomax distribution.

Property 3.1: Let $T_1,\cdots,T_k$ be k continuous random variables that jointly follow the $\mathrm{GML}_k(a;\theta_1,\cdots,\theta_k,l_1,\cdots,l_k)$ distribution as given in (3). Then, the cumulative distribution function of $T_1,\cdots,T_k$ can be computed as

$$F(x_1,\cdots,x_k)=P(T_1\le x_1,\cdots,T_k\le x_k)=P(U_1\le 1,\cdots,U_k\le 1),$$

where $U_1,\cdots,U_k$ jointly follow $\mathrm{GML}_k(a;\theta_1x_1,\cdots,\theta_kx_k,l_1,\cdots,l_k)$ distribution.

Proof of the above is straightforward by making the substitutions $u_i=t_i/x_i$, $i=1,\cdots,k$.

Through parameter substitutions, the cumulative distribution functions of multivariate F and the inverted beta distribution can be found. These are summarized in Table 3.

For the above method, the run-time consumption rapidly increases as k becomes large. Thus as an alternative, we also provide the option of computation of cumulative distribution function via Monte Carlo method. The corresponding algorithm is

Distribution	Parameter Substitutions	Cumulative Distribution Function
Multivariate F with degrees of freedom $(2a, 2l_1, \ldots, 2l_k)$	$\theta_i = l_i/a$, $i = 1, \cdots, k$	$F(x_1, \cdots, x_k) = P(U_1 \leq 1, \cdots, U_k \leq 1)$, $(U_1, \cdots, U_k)$ follows $\text{GML}_k(a; x_1 l_1/a, \cdots, x_k l_k/a, l_1, \cdots, l_k)$
Multivariate Inverted Beta	$\theta_i = 1$, $i = 1, \cdots, k$	$F(x_1, \cdots, x_k) = P(U_1 \leq 1, \cdots, U_k \leq 1)$, $(U_1, \cdots, U_k)$ follows $\text{GML}_k(a; x_1, \cdots, x_k, l_1, \cdots, l_k)$

Table 3: Cumulative distribution functions of related multivariate distributions to generalized multivariate Lomax.

Algorithm - CDF Computation using Monte Carlo Method:

1. Generate N random vectors $\boldsymbol{t}^{(i)} = (t_1^{(i)}, \cdots, t_k^{(i)})', i = 1, \cdots, N$ from the desired distribution;
2. Compute

$$\hat{P}(T_1 \leq x_1, \cdots, T_k \leq x_k) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{I}(\boldsymbol{t}^{(i)} \leq \boldsymbol{x}),$$

where $\boldsymbol{x} = (x_1, \cdots, x_k)'$ and $\mathcal{I}(\cdot)$ is the zero-one indicator function corresponding to the conditions specified.

Step 1 above is readily carried out by the random numbers generation as described in Section Multivariate Lomax and related distributions using the package **NonNorMvtDist**. Since cdf is computable using adaptive multivariate integration over unit cube $[0,1]^k$ or via the Monte Carlo method, it follows that the survival function can also be calculated (by using (6)). The equicoordinate quantile is computed by using (7).

We also add in our package the calculations of joint probability density function - Being self-explanatory with all $pdfs$ available in closed form, it needs no further elaboration. The corresponding function is `dmv*()`.

Illustrations of simulations and probability calculations

We will illustrate here the functions and corresponding arguments for **NonNorMvtDist**. The calling sequences include probability density calculation (`dmv*`), cumulative distribution calculation (`pmv*`), equicoordinate quantile calculation (`qmv*`), random numbers generation (`rmv*`), and survival function calculation (`smv*`) for each of the multivariate distributions introduced in Section Multivariate Lomax and related distributions. For each distribution, we consider the bivariate case ($k = 2$). This choice enables us to also succinctly and graphically present the probability density plots. The detailed description of the calling sequence for each of the several cases has been moved into a digital complement of this paper.

For example, for the bivariate Lomax distribution ($k = 2$) with parameters $a = 5$, $\boldsymbol{\theta} = (0.5, 1)$, the calling sequences for various functions are

```
dmvlomax(x, parm1 = 5, parm2 = (0.5,1))
pmvlomax(q, parm1 = 5, parm2 = (0.5,1))
qmvlomax(p, parm1 = 5, parm2 = (0.5,1))
rmvlomax(n, parm1 = 5, parm2 = (0.5,1))
smvlomax(q, parm1 = 5, parm2 = (0.5,1))
```

It may be mentioned that our approach may be more efficient than the NORTA method (Ghosh and Henderson, 2002) for simulation in that NORTA always first requires simulation from multivariate normal, which are then transformed to multivariate uniform. Only often this step, one could subsequently transform the simulated data to the desired distribution. Consequently, for large dimensions, the approach requires more computing power and time (in fact, to simulate data from the normal distribution, many programs themselves first require random number generations from uniform distributions, from which normal random numbers are obtained).

3D bivariate density plot

For each bivariate distribution, we provide the density surface plots along with contours using the function persp3D() from the package **plot3D** (Soetaert, 2017). To illustrate, we define function dplot2(), and we pass appropriate density functions to the density function argument dfun to create the density surfaces. These are summarized in the digital complement along with corresponding resulting plots. For the Lomax distribution with parameters indicated previously, the statements will be

```
library(plot3D)

dplot2 <- function(dfun, x1, x2, zlim) {
    zmat <- matrix(0, nrow = length(x1), ncol = length(x2))
    for (i in 1:length(x1)) {
        for (j in 1:length(x2)) {
            zmat[i, j] = dfun(x = c(x1[i], x2[j]))
        }
    }
    persp3D(z = zmat, x = x1, y = x2, theta = -60, phi = 10, ticktype = "detailed",
            zlim = zlim, contour = list(nlevels = 30, col = "red"),
            facets = FALSE, image = list(col = "white", side = "zmin"),
            xlab = "X1", ylab="X2", zlab = "Density", expand = 0.5, d = 2)
}

dplot2(dfun = function(x) dmvlomax(x, parm1 = 5, parm2 = c(0.5, 1)), x1 = seq(0, 4, 0.1),
       x2 = seq(0, 4, 0.1), zlim = c(-5, 13))
```

The plot that results is shown in Figure 2.

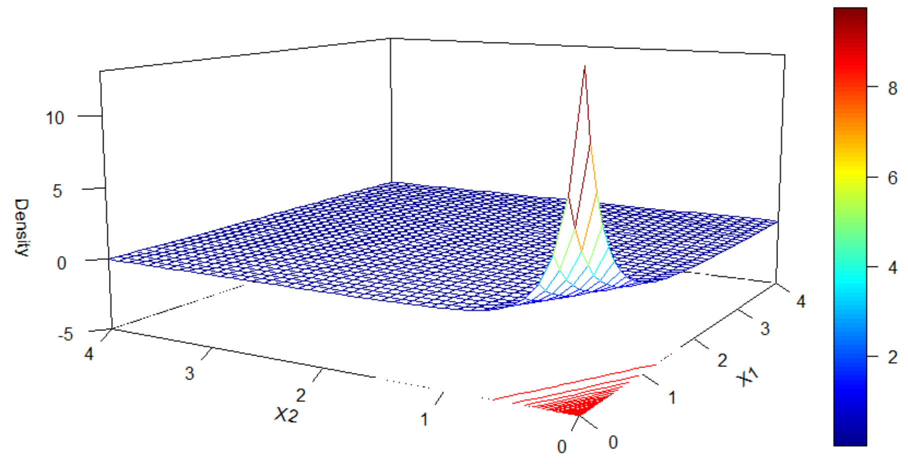


Figure 2: Density surface of bivariate Lomax distribution with parameters $a = 5, \boldsymbol{\theta} = (0.5, 1)$.

Random number generation

The following code illustrates the use of the function rmvlomax*() with a bivariate sample of size $n = 2$. Sampling is done by setting set.seed(2019) in advance. The digital complement explicitly provides the code as well as output for all of the probability distributions discussed here. In the output, each row represents a bivariate observation.

- **Bivariate Lomax:** $a = 5, \theta_1 = 0.5, \theta_2 = 1$

```
> set.seed(2019)
> rmvlomax(n = 2, parm1 = 5, parm2 = c(0.5, 1))
          [,1]      [,2]
[1,] 1.0174406 0.7076480
[2,] 0.3686253 0.7826978
```

CDF, survival function and equicoordinate quantile

The applications of *cdf* `pmv*()`, survival function `smv*()`, and equicoordinate quantile function `qmv*()` are straightforward and follow the same pattern earlier. See the digital complement for computation details. In the following, we give code as well as output only for Lomax distribution ($a = 5$, $\theta_1 = 0.5$, $\theta_2 = 1$) for specified coordinates (x_1, x_2) and for the cumulative probability $p = 0.5$.

- **Bivariate Lomax:** $a = 5, \theta_1 = 0.5, \theta_2 = 1$; quantiles: $(x_1, x_2) = (1, 0.5)$.

```
> pmvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))
[1] 0.7678755
> smvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))
[1] 0.03125
> qmvlomax(p = 0.5, parm1 = 5, parm2 = c(0.5, 1))
[1] 0.3928917
```

Computation times

We assess the run-times for the computation of probability (`pmv*`), equicoordinate quantile function (`qmv*`) for multivariate Lomax, and generalized multivariate Lomax distributions for reference. The survival function (`smv*`) of multivariate Lomax distribution has a closed-form expression, and hence the assessment of computation time is omitted. We have used the computer with Intel Core i5-8250U CPU and 8.00 GB RAM. The results for p-variate Lomax distribution are summarized in Table 4. As we can see, the run-times for `pmvlomax()` are quit short, even for the dimension $p = 20$. However, `qmvlomax()` requires a considerable longer time when $p \geq 17$, which seems to double for every extra dimension added to the size of the random vector. The computation times in Table 4 can also be used as a reference for the distributions related to multivariate Lomax, which are in Table 2 since we apply the same approach for probability computations there as well.

p	`pmvlomax()`	`qmvlomax()`	p	`pmvlomax()`	`qmvlomax()`
1	0.01695895	0.03702188	11	0.04889703	0.7992568
2	0.01795197	0.02293706	12	0.09025002	1.879766
3	0.01794696	0.02789807	13	0.1545889	3.087925
4	0.01795197	0.03290296	14	0.245369	5.950396
5	0.01976085	0.04189205	15	0.4657819	11.37162
6	0.01995182	0.06582808	16	0.853502	22.04591
7	0.02094102	0.098768	17	1.708418	45.51909
8	0.02197218	0.146611	18	2.803703	90.70764
9	0.02396512	0.2393882	19	5.453189	181.55772
10	0.04587412	0.4296861	20	10.40903	351.58896

Table 4: Runtimes (in seconds) for functions `pmvlomax()` and `qmvlomax()` as functions of p.

The results for generalized p-variate Lomax distribution are summarized in Tables 5 and 6. Both functions `pmvglomax()` and `smvglomax()` require relatively much longer time when $p > 4$, and `qmvglomax()` takes longer when $p > 2$. Based on the run-time consumption, we recommend algorithm `MC` for larger dimensions (e.g., when $p > 5$). Similarly, this run-time study can also be used as a reference for the related distributions (to generalized Lomax distribution) as listed in Table 3 since we apply the same method for computations.

	`pmvglomax()`		`smvglomax()`	
p	`numerical`	`MC`	`numerical`	`MC`
1	0.03395414	4.600386	0.03084397	3.977374
2	0.02397585	6.536522	0.05378199	5.510087
3	0.101758	8.396577	0.208086	7.229721
4	0.9758801	10.64278	2.13447	8.741386
5	16.43411	19.30024	40.11997	9.762063
6	305.50092	19.57221	1344.1908	11.63218

Table 5: Runtimes (in seconds) for functions `pmvglomax()` and `smvglomax()` by using algorithms `numerical` and `MC` as functions of p.

	qmvglomax()	
p	numerical	MC
1	0.4144461	12.5594
2	6.383504	18.97528
3	96.86238	27.23917
4	2929.5018	30.62036

Table 6: Runtimes (in seconds) for function `qmvglomax()` by using algorithms `numerical` and `MC` as functions of p.

Maximum likelihood estimation of parameters

We also include the maximum likelihood estimation to estimate the parameters for various Lomax related distributions (except for the bivariate F distribution). Although many of the density functions in Section Multivariate Lomax and related distributions have complicated forms, maximum likelihood estimation can be easily accomplished by using the built-in optimization functions in R **stats**. The log-likelihood function for a given sample $\mathbf{x_1}, \ldots, \mathbf{x_n}$ is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathbf{x_1}, \ldots, \mathbf{x_n}) &= \log\left[\prod_{j=1}^{n} f(x_{j1}, \ldots, x_{jk}|\boldsymbol{\theta})\right] \\ &= \sum_{j=1}^{n} \log\left[f(x_{j1}, \ldots, x_{jk}|\boldsymbol{\theta})\right], \qquad \boldsymbol{\theta} \in \Theta, \end{aligned} \tag{9}$$

where n is the sample size, $\boldsymbol{\theta}$ is a vector of parameters to be estimated, and $\mathbf{x}_j = (x_{j1}, \ldots, x_{jk})'$ is the jth observation for the random vector $\mathbf{X} = (X_1, \ldots, X_k)'$, respectively. The maximizer $\hat{\boldsymbol{\theta}}$ of the log-likelihood function given in (9), namely,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x_1}, \ldots, \mathbf{x_n}),$$

is obtained using an appropriate optimization method. The parameter space Θ in each case must be appropriately constrained, and these constraints must be taken into account during the optimization process. We have thus made use of three R **stats** functions, namely, `optim()`, `constrOptim()`, and `optimize()` in this work. The functionality of these optimization functions is described in Table 7.

Function	**Number of parameters**	**Usage**
`optim()`	Multiple	General-purpose Optimization
`constrOptim()`	Multiple	Optimization with linear constraints
`optimize()`	Single	One Dimensional Optimization

Table 7: Use of optimization functions in R **stats**.

By default, all these functions perform the task of minimization of a function. To maximize (9), we only need to add argument `control = list(fnscale = -1)` in functions `optim()` and `constrOptim()`, and set `maximum = TRUE` in function `optimize()`.

For example, for the multivariate Lomax distribution, we define the log-likelihood function `loglik.lomax()` by using the following code:

```
loglik.lomax <- function(data, par) {
    ll <- sum(dmvlomax(data, parm1 = par[1], parm2 = par[-1], log = TRUE))
}
```

The R **stats** function `constrOptim()` is chosen to obtain the maximizer of the log-likelihood function (or equivalently, `loglik.lomax()`). The linear constraints imposed on the parameters a, θ_1, and θ_2 are $a > 0, \theta_1 > 0$, and $\theta_2 > 0$. In matrix notation, it is,

$$\boldsymbol{U\theta} = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}a\\ \theta_1\\ \theta_2\end{pmatrix} = \begin{pmatrix}a\\ \theta_1\\ \theta_2\end{pmatrix} > \begin{pmatrix}0\\ 0\\ 0\end{pmatrix} = \boldsymbol{C}.$$

Thus, in the code that follows, the constraint matrix `ui` is set as an identity matrix $\mathbf{I}_3$ by using the function `diag(3)`, and constraint vector `ci` is set as a zero vector `rep(0,3)`. For our illustration, let the data be the one simulated by the methods described earlier, each with $n = 300$. It is done by

```
set.seed(1)
bvtlomax <- rmvlomax(n = 300, parm1 = 5, parm2 = c(0.5, 1))
```

The starting initial values for a, θ_1, and θ_2 are all set as 10 by assigning `rep(10,3)` to the argument `theta` in function `constrOptim()`. The gradient argument `grad` is optional, and we have chosen this to be `NULL`.

```
> est = constrOptim(theta = rep(10, 3), f = loglik.lomax, grad = NULL, data = bvtlomax,
+                   ui = diag(3), ci = rep(0, 3), control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 5.0555691 0.4468724 0.9036692
```

The output consists of two important pieces of information, (i) whether convergence is successfully achieved (`est$convergence = 0`) or not (`est$convergence = 1`) and (ii) the values of the final maximum likelihood estimates. In our illustration, the convergence is successfully achieved, and we have $\hat{a} = 5.0555691$ and $(\hat{\theta}_1, \hat{\theta}_2) = (0.4468724, 0.9036692)$.

We summarize the optimization methods, constraints, and the resulting outputs for all the bivariate distributions (except for bivariate F distribution) in Table 8. The detailed illustrations and codes for the remaining distributions are included in the digital complement. Observe that these estimates are reasonably close to the true parameter values, thereby confirming that the program is functioning as it is expected to.

Multivariate Distribution	**Parameters**	**Optimization Method**	**Constraints**	**Estimated Parameters**
Lomax	$a = 5$ $\boldsymbol{\theta} = (0.5, 1)'$	`constrOptim()`	$U = I_3$ $C = (0, 0, 0)'$	$\hat{a} = 5.05556$ $\hat{\boldsymbol{\theta}} = (0.44687, 0.90366)'$
Mardia's Pareto Type I	$a = 5$ $\boldsymbol{\theta} = (0.5, 2)'$	`constrOptim()`	$U = I_3$ $C = (0, [\min(X_1)]^{-1}, [\min(X_2)]^{-1})'$	$\hat{a} = 4.63862$ $\hat{\boldsymbol{\theta}} = (0.49971, 1.99785)'$
Logistic	$\boldsymbol{\mu} = (0.5, 1)'$ $\boldsymbol{\sigma} = (1, 1.5)'$	`optim()`	N/A	$\hat{\boldsymbol{\mu}} = (0.39199, 0.89731)'$ $\hat{\boldsymbol{\sigma}} = (0.97573, 1.55976)'$
Burr	$a = 3$ $\boldsymbol{d} = (1, 3)'$ $\boldsymbol{c} = (2, 5)'$	`constrOptim()`	$U = I_5$ $C = (0, 0, 0, 0, 0)'$	$\hat{a} = 3.91498$ $\hat{\boldsymbol{d}} = (0.64889, 2.06858)'$ $\hat{\boldsymbol{c}} = (1.92719, 5.07697)'$
Cook-Johnson's Uniform	$a = 0.3$	`optimize()`	$a > 0$	$\hat{a} = 0.31064$
Generalized Lomax	$a = 5$ $\boldsymbol{\theta} = (0.5, 1)'$ $\boldsymbol{l} = (2, 4)'$	`constrOptim()`	$U = I_5$ $C = (0, 0, 0, 0, 0)'$	$\hat{a} = 3.91869$ $\hat{\boldsymbol{\theta}} = (0.54674, 1.79126)'$ $\hat{\boldsymbol{l}} = (1.70310, 5.21736)'$
Inverted Beta	$a = 4$ $\boldsymbol{l} = (2, 6)'$	`constrOptim()`	$U = I_3$ $C = (0, 0, 0)'$	$\hat{a} = 3.67153$ $\hat{\boldsymbol{l}} = (1.82450, 5.46465)'$

Table 8: Comparison between True and Estimated Values of Parameters for each of the Distributions (except for bivariate F distribution).

Two applications

In this last section, we give two brief applications, which not only demonstrate the use but also confirm the accuracy and verify the correctness of our work.

Generating data from the nonelliptical symmetric distributions with univariate normal marginals

Cook-Johnson's multivariate uniform distribution is a family of distributions that can be used for modeling nonelliptical symmetric data. Further, in view of uniform distribution for marginal, it has been as one of the useful choices for modeling through copula (in fact, Cook-Johnson's uniform distribution is indeed a Clayton copula (Nelsen, 2006)). The value of parameter a impacts the common correlation coefficient ρ among variates in that $\rho \to 0$ as $a \to \infty$, and $\rho \to 1$ as $a \to 0$ (Cook and Johnson, 1981). An interesting application of Cook-Johnson's multivariate uniform distribution is to obtain new joint distributions by marginal transformation. Specifically, we consider the problem of generating random numbers from a multivariate distribution that is not elliptically symmetric but has univariate normal marginals. Let U_i's, $i = 1, 2$, be two random variables corresponding to the Cook-Johnson's bivariate uniform distribution. The following code yields the pairs of random numbers, each having the normal marginals by the transformation $X_i = \Phi^{-1}(Ui)$, where $\Phi^{-1}(\cdot)$ is the quantile function of a standard normal distribution. Clearly, the joint distribution of X_1 and X_2 is not bivariate normal. To begin with, the parameter a is taken to be $a = 2$.

```
set.seed(1)
biv.unif <- rmvunif(8000, parm = 2, dim = 2)
biv.norm <- as.data.frame(apply(biv.unif, 2, qnorm))
```

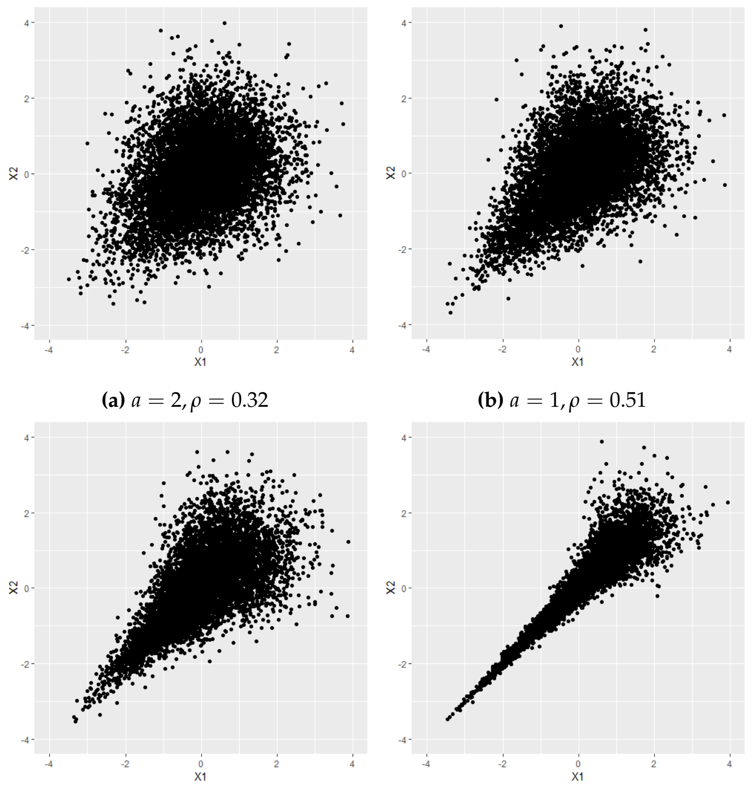
The sample correlation coefficient ρ of data set `biv.norm` is computed by the following code.

```
> cor(biv.norm$V1, biv.norm$V2)
[1] 0.3180119
```

We create a bivariate scatter plot using the function `ggplot()` in package **ggplot2** (Wickham, 2016) for data set `biv.norm`. This is shown in Figure 3 (a).

```
library(ggplot2)

ggplot(biv.norm, aes(x = V1, y = V2)) + xlim(c(-4, 4)) +
ylim(c(-4, 4)) + xlab("X1") + ylab("X2") + geom_point()
```

To assess the behavior as a function of a, we now decrease the parameter a to $1.0, 0.5, 0.1$ resulting in higher correlations ρ $(= 0.51, 0.68, 0.93$, respectively) between the two variates. The bivariate scatter plots for the four cases that is, when $a = 2.0, 1.0, 0.5, 0.1$ are shown in Figure 3 (a)-(d). It is easy to observe that the generated bivariate data have nonelliptical yet, symmetric contours.



(a) $a = 2, \rho = 0.32$ **(b)** $a = 1, \rho = 0.51$

(c) $a = 0.5, \rho = 0.68$ **(d)** $a = 0.1, \rho = 0.93$

Figure 3: The scatterplots of nonelliptical symmetric normal data generated from transformed Cook-Johnson's uniform random numbers with $a = 2, 1, 0.5$ and 0.1.

Creating tables for simultaneous MANOVA hypothesis tests

Multivariate F distribution arises naturally as the distribution of test statistics in several testing problems in simultaneous MANOVA. Let $s_1^2, \cdots, s_k^2$ be k independent sums of squares, and s_0^2 be the sum of squares due to error in the classical ANOVA model. Also, let $H_1, \cdots, H_k$ be certain individual linear hypotheses with the corresponding sum of squares $s_1^2, \cdots, s_k^2$. Assume that under $H_1, \cdots, H_k$, respectively, the sums of squares $s_1^2, \cdots, s_k^2$ are all χ^2 random variables each with n degrees of freedom and s_0^2 is a χ^2 random variable with m degrees of freedom and is independent of $s_1^2, \cdots, s_k^2$. Armitage and Krishnaiah (1964) defined the critical values F_α at α level of significance for simultaneously testing hypotheses $H_1, \cdots, H_k$ by the probability statement,

$$P\left[\frac{m \max\left(s_1^2, \cdots, s_k^2\right)}{n s_0^2} \le F_\alpha \,\middle|\, \bigcap_{i=1}^{k} H_i\right] = P\left[\frac{m s_i^2}{n s_0^2} \le F_\alpha, i = 1, \cdots, k \,\middle|\, \bigcap_{i=1}^{k} H_i\right] = 1 - \alpha.$$

The quantities $(ms_i^2)/(ns_0^2), i = 1, \cdots, k$ jointly follow multivariate F distribution if the overall null hypothesis $H_0 = \bigcap_{i=1}^k H_i$ is true. In this case, the critical value F_α can be readily computed using the equicoordinate quantile function `qmvf()` by setting the argument corresponding to $k+1$ values of the degrees of freedom as `df = c(m,n,...,n)`. The following code gives $F_{0.05} = 9.551505$ for the bivariate F case when $m = 5$ and $n = 1$ with default algorithm by using adaptive multiple integration over unit cube (`algorithm = "numerical"`). With Monte Carlo algorithm (`algorithm = "MC"` with `nsim=1,000,000`), we obtain $F_{0.05} = 9.550944$. Note that the Monte Carlo method is seed dependent, so the output from different runs may slightly differ from each other.

```
> qmvf(0.95, df = c(5, 1, 1))
[1] 9.551505
> qmvf(0.95, df = c(5, 1, 1), algorithm = "MC")
[1] 9.550944
```

For further demonstration and also to further affirm our trust in the calculations, we compare the output of quantile function `qmvf()` using both adaptive multivariate integration and Monte Carlo methods with the values given in Armitage and Krishnaiah (1964). These three calculations are reported in Table 9 for a few choices of m and n. The agreement among the three columns shows that the package **NonNorMvtDist** provides a convenient way to obtain percentage points for the hypothesis testing problems considered by Armitage and Krishnaiah (1964) and Krishnaiah (1965). Clearly, unlike the tables in Armitage and Krishnaiah, the choices of α and degrees of freedom are not restricted, and in that sense, our package is very comprehensive and exhaustive in this respect.

α	**df** (m,n,n)	`qmvf()` **Output** (`algorithm = "numerical"`)	`qmvf()` **Output** (`algorithm = "MC"`)	**Tabulated Values in Armitage and Krishnaiah (1964, pp. 33-42)**
0.05	$(5,1,1)$	9.551505	9.550944	9.55
	$(5,2,2)$	7.879999	7.881698	7.88
	$(5,3,3)$	7.136473	7.165361	7.14
	$(5,4,4)$	6.702224	6.715759	6.70
	$(5,5,5)$	6.412372	6.399167	6.41
	$(10,6,6)$	3.899335	3.898442	3.90
	$(10,7,7)$	3.768494	3.767915	3.77
	$(10,8,8)$	3.665646	3.661366	3.67
	$(10,9,9)$	3.582271	3.583923	3.58
	$(10,10,10)$	3.513163	3.514059	3.51

Table 9: Comparison between the output of `qmvf()` with the values given in Armitage and Krishnaiah (1964).

Concluding remarks

We have developed a new R package, **NonNorMvtDist**, for generating multivariate random numbers from Lomax (Pareto type II), generalized Lomax, Mardia's Pareto type I, logistic, Burr, Cook-Johnson's uniform, F, and inverted beta distributions. Detailed examples of each distribution are given to illustrate data simulation, probability calculations, and statistical modeling.

The fact that nonnormal and skewed multivariate distributions are common in the real world but are rarely pursued for analysis due to the lack of ready-to-use computational support underscores the importance of this package. Possibilities of the use of these distributions are practically limitless and yet unforeseen in a variety of areas, starting from the biomedical sciences, reliability, and engineering as well as in statistical finance in the contexts of volatility estimation. Simulations, probability calculations, as well as calculations of quantiles, and the maximum likelihood estimation of parameters are the natural first set of computations in such studies. We have addressed all of these in this work.

The calculations of probabilities of hypercubes (for example, of $P[a_1 < X_1 < b_1, a_2 < X_2 < b_2, a_3 < X_3 < b_3]$) can be easily implemented by appropriately combining several cdf calculations. Alternatively, our codes for `pmv*()` can be suitably modified for this purpose. The probability density surface plots for *any* bivariate marginal can be easily constructed since, for the multivariate Lomax distribution, the marginal distributions of any subset of random variables also follow the multivariate Lomax distribution in the appropriate dimension. Further, our work provides a way to generate data from, probability calculations for, as well as modeling for, the data which are marginally distributed as normal but jointly are not.

Acknowledgements

Content of this article constitutes a part of the PhD dissertation of Zhixin Lun at Oakland University. Financial research support from Oakland University during the summer of Year 2019 is acknowledged. We would like to thank the anonymous referees for their helpful comments and remarks to improve both package and this paper.

Bibliography

J. V. Armitage and P. R. Krishnaiah. *Tables for Studentized Largest Chi-Square and Their Application*. Report ARL 64-188, Aerospace Research Laboratories Wright-Patterson Air Force Base, Ohio, 1964. [p429, 438]

N. Balakrishnan and C.-D. Lai. *Continuous Bivariate Distributions*. Springer-Verlag New York, 2nd edition, 2009. ISBN 978-0-387-09614-8. URL `https://doi.org/10.1007/b101765`. [p428]

R. E. Cook and M. E. Johnson. A family of distributions for modeling non-elliptically symmetric multivariate data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 43(2):210–218, 1981. URL `www.jstor.org/stable/2984851`. [p437]

S. Ghosh and S. Henderson. Properties of the norta method in higher dimensions. *Proceedings of the 2002 Winter Simulation Conference*, 1:263–269, 2002. URL `https://doi.org/10.1109/WSC.2002.1172894`. [p432]

H. Joe. *Multivariate Models and Dependence Concepts*. Chapman and Hall, London, 1997. URL `https://doi.org/10.1201/b13150`. [p430]

N. L. Johnson and S. Kotz. *Distribution in Statistics: Continuous Multivariate Distributions*. John Wiley & Sons, New York, 1972. [p429]

P. R. Krishnaiah. On the simultaneous anova and manova tests. *Annals of the Institute of Statistical Mathematics*, 17(1):35–53, 1965. URL `https://doi.org/10.1007/BF02868151`. [p429, 438]

D. V. Lindley and N. D. Singpurwalla. Multivariate distributions for the life lengths of a system sharing a common environment. *Journal of Applied Probability*, 23:418–431, 1986. URL `https://doi.org/10.2307/3214184`. [p427]

Z. Lun and R. Khattree. *NonNorMvtDist: Multivariate Lomax (Pareto Type II) and Its Related Distributions*, 2020. URL `https://CRAN.R-project.org/package=NonNorMvtDist`. R package version 1.0.2. [p427]

B. Narasimhan, S. G. Johnson, T. Hahn, A. Bouvier, and K. Kiêu. *cubature: Adaptive Multivariate Integration over Hypercubes*, 2018. URL `https://CRAN.R-project.org/package=cubature`. R package version 2.0.3. [p431]

T. K. Nayak. Multivariate lomax distribution: Properties and usefulness in reliability theory. *Journal of Applied Probability*, 24(1):170–177, 1987. URL `https://doi.org/10.2307/3214068`. [p427, 428, 429]

R. B. Nelsen. *An Introduction to Copulas*. Springer-Verlag New York, 2nd edition, 2006. ISBN 978-0-387-28659-4. URL `https://doi.org/10.1007/0-387-28678-0`. [p436]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org/. [p429]

K. Soetaert. *plot3D: Plotting Multi-Dimensional Data*, 2017. URL https://CRAN.R-project.org/package=plot3D. R package version 1.1.1. [p433]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p437]

T. Yee. *VGAM: Vector Generalized Linear and Additive Models*, 2019. URL https://CRAN.R-project.org/package=VGAM. R package version 1.1-2. [p427]

Zhixin Lun
Department of Clinical Pharmacy
University of California, San Francisco
San Francisco, CA, USA
ORCiD: 0000-0002-8980-1554
zhixin.lun@ucsf.edu

Ravindra Khattree
Department of Mathematics and Statistics
Oakland University
Rochester, MI, USA
ORCiD: 0000-0002-9305-2365
khattree@oakland.edu

CompModels: A Suite of Computer Model Test Functions for Bayesian Optimization

by Tony Pourmohamad

Abstract The **CompModels** package for R provides a suite of computer model test functions that can be used for computer model prediction/emulation, uncertainty quantification, and calibration. Moreover, the **CompModels** package is especially well suited for the sequential optimization of computer models. The package is a mix of real-world physics problems, known mathematical functions, and black-box functions that have been converted into computer models with the goal of Bayesian (i.e., sequential) optimization in mind. Likewise, the package contains computer models that represent either the constrained or unconstrained optimization case, each with varying levels of difficulty. In this paper, we illustrate the use of the package with both real-world examples and black-box functions by solving constrained optimization problems via Bayesian optimization. Ultimately, the package is shown to provide users with a source of computer model test functions that are reproducible, shareable, and that can be used for benchmarking of novel optimization methods.

Introduction

The **CompModels** package (Pourmohamad, 2020) for R (R Core Team, 2020) is a suite of test functions designed to mimic computer models. Usually deployed when physical experimentation is not possible, a computer model (or code) is a mathematical model that simulates a complex phenomena or system under study via a computer program. For example, weather phenomena, such as hurricanes or global warming, are not reproducible physical experiments. Therefore, computer models based on climatology are used to study these events. At its simplest, a computer model is a mathematical model of the form

$$y = f(x_1, \ldots, x_d) = f(x), \quad x = (x_1, \ldots, x_d)^T \in \mathcal{X}, \tag{1}$$

where x is an input variable to the computer model, y is a (possibly multivariate) deterministic output from the computer model, and $\mathcal{X}$ is the domain of the input variable. A defining characteristic of most computer models is that, for a given input x, the evaluation of the underlying mathematical model, f, is a time intensive endeavor. Computationally expensive computer models helped spur the development of the computer modeling field in statistics (Santner et al., 2003), and in particular, the development of "cheap-to-compute" statistical models, or *surrogate models*, that resemble the true computer model very closely but are much faster to run. Outside the scope of this paper, but useful for forthcoming discussion and illustrations, we simply mention that Gaussian processes (GPs) (Stein, 1999) have been used as the typical modeling choice for building statistical surrogate models. GPs are the preferred choice of statistical surrogate model due to their flexibility, well-calibrated uncertainty, and analytic properties (Gramacy, 2020).

Another typical trait of computer models is that they are often treated as black-box functions. Here, a black-box computer model is a computer model where evaluation requires running computer code that reveals little information about the functional form of the underlying mathematical function, f. The black-box assumption often arises due to the fact that f may be extremely complex, analytically intractable, or that access to the internal workings of the computer model are restricted, say, for such reasons as being proprietary software. The latter restricted cases have led to a dearth of real-world computer models that are freely available and/or accessible to statisticians that hope to develop novel methods for the computer modeling field. It is for this reason that we have developed the **CompModels** package which serves as a repository of pseudo computer models for statistical use.

The **CompModels** package can be used to test and develop methods for computer model emulation (prediction), uncertainty quantification, and calibration. However, the main focus when developing the package was placed on building computer models for optimization. Real-world computer models are often built with the goal of understanding some physical system of interest, and with that goal usually comes the need to optimize some output of interest from the computer model. For example, in hydrology, the minimization of contaminants in rivers and soils is of interest and so computer models representing pump-and-treat remediation plans are often used in order to optimize objectives, such as the associated costs of running pumps for pump-and-treat remediation, while also ensuring that contaminants do not spread (Pourmohamad and Lee, 2019). Recalling that most computer models are computationally expensive to run, the need for efficient sequential optimization algorithms (also known as Bayesian optimization) that do not require many functional evaluations is high, which is why the focus of the test functions in the **CompModels** package is placed on optimization. More specifically, the **CompModels** package presents functions to optimize of the following form

$$\min_{x}\{f(x) : c(x) \leq 0,\, x \in \mathcal{X}\}, \tag{2}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is a known, bounded region such that $f : \mathcal{X} \rightarrow \mathbb{R}$ denotes a scalar-valued objective function, and $c : \mathcal{X} \rightarrow \mathbb{R}^m$ denotes a vector of m constraint functions. However, many of the package functions omit the constraint functions and thus the package is a mix of constrained and unconstrained optimization problems.

Some of the functions in the **CompModels** package have known functional forms, for example, the `gram()` and `mtp()` functions. However, most all functions are intended to serve as black-box computer models. All of the black-box computer model functions within the package are aptly named `bbox` (short for black-box) and followed by a unique integer value to make the functions discernible. For example, `bbox1()` and `bbox2()` are two unique function calls to two different black-box computer models that can be used for constrained and unconstrained optimization, respectively. R is an open-source programming language, and so none of the computer models within the package can ever truly be a completely black-box function. However, the developers of the **CompModels** package have done their best to obscure the analytical forms of the mathematical functions underlying the computer models. For example, at the first level of the code, a call to the `bbox1()` function tells the user the following:

```
R> bbox1
function(x1,x2){

  if(!is.numeric(x1) | !is.numeric(x2) | length(x1) != 1 | length(x2) !=1){
    stop("Input is invalid.")
  }else if(x1 < -1.5  | x1 > 2.5 | x2 < -3 | x2 > 3){
    stop("Input is outside of the domain.")
  }else{
    ans <- .C("bbox1c",x1=x1,x2=x2,fx=0,c1x=0,c2x=0)
    return(list(obj = ans$fx, con = c(ans$c1x,ans$c2x)))
  }
}
```

The only discernible information that the user can glean from this output is that the `bbox1()` function has an input dimension of $d = 2$, where the domain $\mathcal{X} = [-1.5, 2.5] \times [-3, 3]$, and that there is one objective function, `fx`, to minimize, and two constraint functions, `c1x` and `c2x`, to satisfy. As we see from the `.C()` command, the actual source code for the black-box function has been written using the C programming language. The C programs are publicly available, but the code within those programs has been heavily obfuscated to the best of our abilities in order to obscure the source code such that the computer models remain black-box functions. Moreover, we believe that a good robust methodology developed for computer models benefits from being applied to black-box functions and so any attempt to decipher the black-box computer models is simply a disservice to the statistician developing the methodology.

When developing the computer models in the package, we kept in mind that the best computer model examples typically have roots in real applications. When possible, we tried to develop computer models that were either based on physics or that appeared in the literature with real use cases. For example, one computer model, `pressure()`, is based on the real-world engineering problem of minimizing the cost associated with constructing a pressure vessel (Figure 1). Given the thickness of the shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4) not including the head, the cost of constructing the pressure vessel is to be minimized subject to four constraints on the cost of materials, forming, and welding.

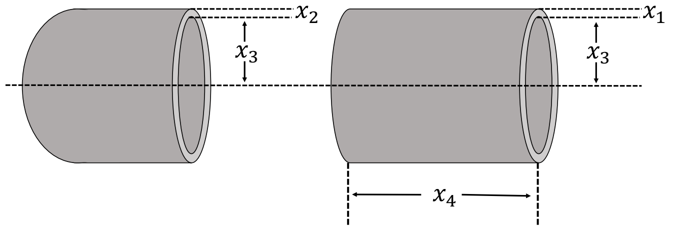


Figure 1: The physical representation of the pressure vessel computer model.

Likewise, when possible, we sought out real-world problems where solutions already existed that could be benchmarked to. For example, the tension spring computer model, `tension()`, is designed to minimize the weight of a tension spring (Figure 2) subject to four constraints on the shear stress, surge frequency, and deflection. The three inputs to the computer model are for the wire diameter (x_1), mean coil diameter (x_2), and the number of active coils (x_3).

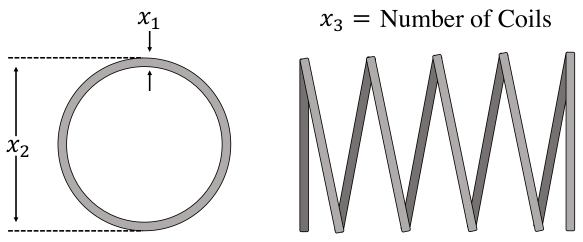


Figure 2: The physical representation of the tension spring computer model.

The tension spring problem has been solved many times in the literature, and Table 1 summarizes some of the best solutions.

Source	Optimal Inputs			Best Solution
	x_1	x_2	x_3	
Coello (2000)	0.051480	0.351661	11.632201	0.012704
He and Wang (2007)	0.051728	0.357644	11.244543	0.012675
Gandomi et al. (2013)	0.051690	0.356730	11.288500	0.012670
Mirjalili et al. (2014)	0.051690	0.356737	11.288850	0.012666
Lee and Geem (2005)	0.051154	0.349871	12.076432	0.012671
Askarzadeh (2016)	0.051689	0.356717	11.289012	0.012665
Mirjalili et al. (2017)	0.051207	0.345215	12.004032	0.012676
Li et al. (2019)	0.051618	0.355004	11.390144	0.012665

Table 1: Best solutions to the tension spring optimization problem from the literature.

We stress the need for benchmarking in our examples because we believe that benchmarking also helps with allowing for good computer model methodology to be developed. In the computer modeling literature, one tends to see real-world optimization results that stand alone and cannot be compared against or even replicated because practitioners do not have access to the same computer models as others. Being able to benchmark one's results to others helps discern how well a given optimization method performs and allows for useful internal feedback when developing a method. Thus, a key reason we have developed the **CompModels** package is so that equitable access to computer models for benchmarking exists. Similarly, a problem with real-world computer models is that they can change over time, and often older versions will be phased out, unsupported, or disappear entirely. For example, the optimization results for the MODFLOW-96 computer model (McDonald and Harbaugh, 1996) from Pourmohamad and Lee (2016) was benchmarked to the work in Lindberg and Lee (2015). However, this computer model is no longer supported by its developers, and so future benchmarking may become infeasible. Thus, the **CompModels** package also stands as a repository of computer models that should be available to all users for the foreseeable future. Lastly, computer models can often be platform and operating system specific, which ultimately limits the number of potential users of the computer model. Given that R packages, for the most part, tend to be immune to this problem, the **CompModels** package would be available to as wide of an audience as possible, again providing equitable access to computer models.

The remainder of the paper is organized as follows. Section 2.2 gives a brief introduction to Bayesian optimization and expected feasible improvement so that the computer models within the **CompModels** package can be demonstrated. Section 2.3 illustrates practical applications of package use for optimization, and Section 2.4 concludes with a discussion.

Bayesian Optimization

Tracing its roots as far back as to Mockus et al. (1978), Bayesian optimization (BO) is a sequential design strategy for efficiently optimizing black-box functions in a few steps that does not require gradient information (Brochu et al., 2010). More specifically, BO seeks to solve the minimization problem

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x). \tag{3}$$

The minimization problem in (3) is solved by iteratively developing a statistical surrogate model of the unknown objective function f, and at each step of this iterative process, using predictions from the statistical surrogate model to maximize an acquisition (or utility) function, $a(x)$. The role of the acquisition function is to measure how promising each location in the input space, $x \in \mathcal{X}$, is if it were to be the next chosen point to evaluate. As alluded to in Section 2.1, the GP is the typical choice of surrogate model in the computer modeling literature, and so we adopt that stance as well in this paper. Lastly, although the general definition of BO is that of an unconstrained optimization problem, extensions to the constrained optimization case are straightforward and many (Lee et al., 2011; Gramacy et al., 2016; Letham et al., 2019; Pourmohamad and Lee, 2020). Here, we merely augment the original problem statement in (3) to be

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x) \;\text{ subject to }\; c(x) \leq 0, \tag{4}$$

where now both f and c can be modeled using independent GPs, and all other steps proceed as before.

In order to solve the problems in (3) and (4), an acquisition function must be chosen for efficiently guiding the search. Perhaps, one of the most popular acquisition functions for unconstrained Bayesian optimization is that of expected improvement (EI) (Jones et al., 1998). Originally introduced in the computer modeling literature, Jones et al. (1998) defined the improvement statistic at a proposed input x to be $I(x) = \max_x\{0, f^n_{\min} - Y(x)\}$, where, after n runs of the computer model, $f^n_{\min} = \min\{f(x_1), ..., f(x_n)\}$ is the current minimum value observed. Since the proposed input x has not yet been observed, $Y(x)$ is unknown and can be regarded as a random variable. Likewise, $I(x)$ can be regarded as a random variable, and so new candidate inputs, x^*, can be selected by maximizing the expected improvement, i.e.,

$$x^* = \arg\max_{x \in \mathcal{X}} \mathbb{E}[I(x)]. \tag{5}$$

Fortunately, if we treat $Y(x)$ as coming from a GP then, conditional on a particular parameterization of the GP, the EI acquisition function is available in closed form as

$$\mathbb{E}[I(x)] = (f^n_{\min} - \mu^n(x))\Phi\left(\frac{f^n_{\min} - \mu^n(x)}{\sigma^n(x)}\right) + \sigma^n(x)\phi\left(\frac{f^n_{\min} - \mu^n(x)}{\sigma^n(x)}\right). \tag{6}$$

Here, $\mu^n(x)$ and $\sigma^n(x)$ are the mean and standard deviation of the predictive distribution of $Y(x)$, and $\Phi(\cdot)$ and $\phi(\cdot)$ are the standard normal cdf and pdf, respectively.

Extending EI to the constrained optimization case, Schonlau et al. (1998) defined expected feasible improvement (EFI) as

$$\text{EFI}(x) = \mathbb{E}[I(x)] \times \Pr(c(x) \leq 0), \tag{7}$$

where $\Pr(c(x) \leq 0)$ is the probability of satisfying the joint constraints. Here, $I(x)$ uses an $f^n_{\min}$ defined over the region where the constraint functions are satisfied. Again, new candidate inputs, x^*, can now be selected by maximizing the expected feasible improvement, i.e.,

$$x^* = \arg\max_{x \in \mathcal{X}} \mathbb{E}[I(x)] \times \Pr(c(x) \leq 0). \tag{8}$$

Here the formula in (6) still holds. However, we are now weighting EI by the probability that x is feasible.

Illustrations

We illustrate the use and functionality of the computer models in the **CompModels** package by solving two constrained optimization problems using the EFI method outlined in Section 2.2. We optimize the tension spring computer model, `tension()`, as well as the black-box 1 computer model, `bbox1()`. In both cases, we perform Monte Carlo experiments where we repeat the optimization routine a total of 30 times to judge the robustness of the solutions. We take advantage of the function `optim.efi()` in the **laGP** package (Gramacy, 2016) for running the EFI algorithm. A full list of the available computer models in the **CompModels** package is given in the Appendix and is generalizable to the proceeding examples.

Tension Spring Computer Model

The goal of the tension spring computer model is to minimize the weight of the tension spring subject to four constraints on the shear stress, surge frequency, and deflection. Here, the inputs to the tension spring computer model are the wire diameter (x_1), mean coil diameter (x_2), and the number of active coils (x_3), where $x_1 \in [0.05, 2]$, $x_2 \in [0.25, 1.3]$, and $x_3 \in [2, 15]$. To evaluate the computer model at a given input, a user needs to supply an input within the given domain, i.e.,

```
R> tension(x1 = 1, x2 = 1, x3 = 3)
$obj
[1] 5

$con
[1]   0.9999582 -45.8166667  -0.9995655   0.3333333
```

All of the computer model functions in the package will return a list where the first element in the list is the value of the objective function, and (in the case of constrained optimization) the second element contains the values of the constraint functions. Here we see that for a wire diameter of `x1 = 1`, mean coil diameter of `x2 = 1`, and `x3 = 3` active coils, that the weight of the tension spring is five. However, the first and last constraint has not been satisfied since those values of `$con` are non-negative. Thus, the input is not a feasible solution to the problem. The input of $x = (1, 1, 3)$ was merely a guess for illustrative purposes. A more reasonable approach to minimizing the tension spring computer model would be to employ the EFI method in Section 2.2. In order to do so, we make use of the function `optim.efi()` in the **laGP** package. To be able to use the `optim.efi()` function, we need to first build a wrapper function (which we call bbox) for our `tension()` function that conforms to the specifications of the `optim.efi()` function.

```
R> bbox <- function(X){
+  output = tension(X[1], X[2], X[3])
+  return(list(obj = output$obj, c = output$con))
}
```

Next, we need to create a matrix that encodes the domain of the computer model inputs.

```
R> B <- matrix(c(.05, .25, 2, 2, 1.3, 15), nrow=3)
```

We can implement the EFI algorithm by passing our wrapper function and domain variable as arguments to the `optim.efi()` function, and then by checking the regions where the solution satisfies the constraints.

```
R> ans <- optim.efi(bbox, B, fhat = TRUE, start = 10, end = 300)
R> constraint <- ifelse(apply(ans$C, 1, max) > 0, "Not Met", "Met")
```

Here, we see that the `optim.efi()` function started with a random input of 10 data points and sequentially chose 290 more inputs for a total of 300 evaluations. The output of `optim.efi()` is a large list storing all steps of the EFI algorithm. We create the `constraint` variable in order to be able to find where the minimum feasible value exists.

```
R> min(ans$obj[constraint == "Met"])
[1] 0.0112376

R> ans$X[ans$obj == min(ans$obj[constraint == "Met"])]
[1] 0.05345441 0.45253754 6.69064005
```

Here, we see that the best feasible value found by the EFI algorithm is at a weight of 0.0112376, which occurs at an input of $x = (0.05345441, 0.45253754, 6.69064005)$. Interestingly, this minimum value found of 0.0112376 is much smaller than all of the best minimums found in our review of the literature (Table 1). To evaluate the robustness of the EFI algorithm for the tension spring computer model, we conduct a Monte Carlo experiment where we repeat the optimization routine 30 times based on different starting input data sets of size 10.

```
R> S <- 30
R> results <- rep(NA, S)
R> for(i in 1:S){
+  ans <- optim.efi(bbox, B, fhat = TRUE, start = 10, end = 300)
+  constraint <- ifelse(apply(ans$C, 1, max) > 0, "Not Met", "Met")
+  results[i] <- min(ans$obj[constraint == "Met"])
+}
```

```
R> summary(results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01081 0.01255 0.01302 0.01325 0.01386 0.01859
```

From the summary of the results, we see that over the 30 Monte Carlo experiments that the EFI algorithm was not able to reliably find as good of a solution over the 300 computer model evaluations. The mean value over the 30 runs was 0.01325, which was much higher than the best solutions presented in Table 1. However, we do see from the summary that the EFI algorithm was able to find at least one more better solution, as compared to the literature, at a spring weight of 0.01081.

Black-box Computer Model

Recalling Section 2.1, the bbox1() computer model has an input dimension of $d = 2$, where the domain $\mathcal{X} = [-1.5, 2.5] \times [-3, 3]$, and that there is one objective function, fx, to minimize, and two constraint functions, c1x and c2x, to satisfy. We can, once again, use the optim.efi() function to perform the EFI algorithm by creating an appropriate wrapper function and domain variable.

```
R> bbox <- function(X){
+  output = bbox1(X[1], X[2])
+  return(list(obj = output$obj, c = output$con))
+}

R> B <- matrix(c(-1.5, -3, 2.5, 3), nrow = 2)
```

We initialize the optim.efi() function with an input data set of 10 points and continue to sequentially evaluate the bbox1() function for a total of 100 input points.

```
R> ans <- optim.efi(bbox, B, fhat = TRUE, start = 10, end = 100)
R> constraint <- ifelse(apply(ans$C, 1, max) > 0, "Not Met", "Met")
```

Checking the EFI algorithm results in the areas where the constraint functions were satisfied, we obtain a best feasible minimum objective function value of -4.61008, which occurs at $x = (0.204649, 2.072964)$.

```
R> min(ans$obj[constraint == "Met"])
[1] -4.610088

R> xbest <- ans$X[ans$obj == min(ans$obj[constraint == "Met"])]
R> xbest
[1] 0.204649 2.072964
```

Now, since the bbox1() function is a black-box computer model, we do not have any analytical way of checking whether or not our solution to the optimization problem is a good one. However, the functions in the **CompModels** package were not developed with the intent of forcing them to be computationally expensive if they need not be. Thus, with an input dimension of $d = 2$, it is very easy to evaluate the bbox1() function on a very dense grid to understand what the potential surface of the objective and constraint functions look like. Doing so does not guarantee us analytically that our solution is a good one, but we will be able to tell visually whether or not our solution is a good one. Plotting the objective and constraint surfaces, we obtain the following (Figure 3).

```
R> n <- 200
R> x1 <- seq(-1.5, 2.5, len = n)
R> x2 <- seq(-3, 3, len = n)

R> x <- expand.grid(x1, x2)
R> obj <- rep(NA, nrow(x))
R> con <- matrix(NA, nrow = nrow(x), ncol = 2)

R> for(i in 1:nrow(x)){
+  temp <- bbox1(x[i,1], x[i,2])
+  obj[i] <- temp$obj
+  con[i,] <- temp$con
+}

R> y <- obj
R> y[con[,1] > 0 | con[,2] > 0] <- NA

R> z <- obj
```

```
R> z[!(con[,1] > 0 | con[,2] > 0)] <- NA

R> par(ps=15)
R> plot(0, 0, type = "n", xlim = c(-1.5, 2.5), ylim = c(-3, 3),
+     xlab = expression(x[1]), ylab = expression(x[2]), main = "Black-box Function")
R> c1 <- matrix(con[,1], ncol = n)
R> contour(x1, x2, c1, nlevels = 1, levels = 0, drawlabels = FALSE, add = TRUE,
+     lwd = 2)
R> c2 <- matrix(con[,2], ncol = n)
R> contour(x1, x2, c2, nlevels = 1, levels = 0, drawlabels = FALSE, add = TRUE,
+    lwd = 2, lty = 2)
R> contour(x1, x2, matrix(y, ncol = n), nlevels = 10, add = TRUE, col = "forestgreen")
R> contour(x1, x2, matrix(z, ncol = n), nlevels = 20, add = TRUE, col = 2, lty = 2)
R> points(xbest[1], xbest[2], pch = 21, bg = "deepskyblue")
```

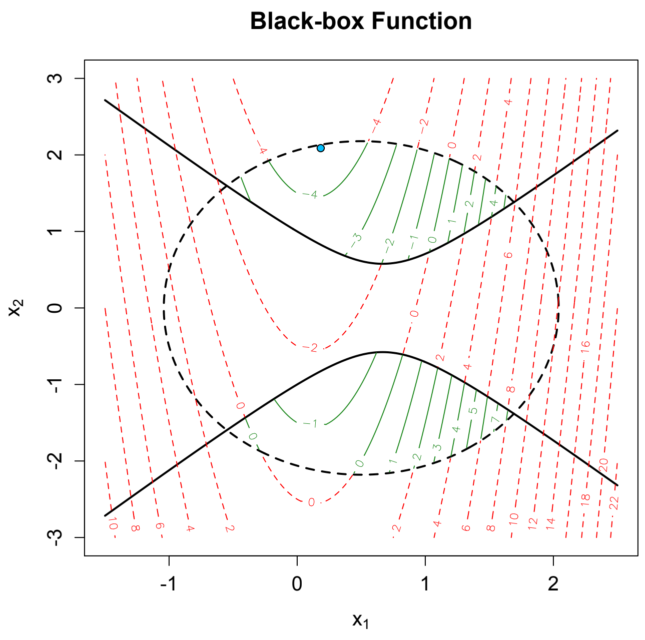


Figure 3: The objective function colored by the two constraints. The solid black line denotes one constraint function, while the dashed black line denotes the other constraint function. Contours that are red are areas where the constraints are not satisfied, while green contours indicate areas where the constraints are satisfied. The blue point represents the best feasible solution found by EFI.

By plotting the objective function surface, and the constraint functions, we see that the space where the constraints are satisfied are two disconnected regions where the feasible region with $x_1 > 0$ has much lower objective function values than the feasible region where $x1 < 0$. We plotted our best minimum objective value found, by EFI, as a blue circle in (Figure 3). Visually, our best minimum objective value found appears to be around the global minimum value based on the calculated contour lines of the plot. Although this visual inspection suggests that our EFI algorithm has correctly identified the global solution to the optimization problem, confirmation of our solution could come from others using the **CompModels** package in order to benchmark the solution. Lastly, we check the robustness of the solution found by the EFI algorithm by conducting a Monte Carlo experiment where we repeat the optimization routine for a total of 30 times.

```
R> summary(results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -4.681  -4.661  -4.645  -4.645  -4.632  -4.602
```

From the summary of the results, we see that the variation in the results show up in the hundredth decimal point and beyond, which we regard as representing a very robust solution.

Discussion

The primary goal of the package is to provide users with a source of computer model test functions that are reproducible, shareable, and that can ultimately be used for benchmarking of Bayesian optimization methods. The package will greatly benefit those who do not have access, or connections, to real-world computer models. In time, it is our hope that the package will come to be viewed as a suite of real computer models rather than solely as pseudo ones. Likewise, the **CompModels** package is not a static package in that we envision it to be a living repository, and so more computer model functions will be expected to be added over time. The success of any R package ultimately comes from the feedback received from its users. We greatly encourage all interested users of the package to please contact the developers in order to provide any insights or examples for new computer models to be added.

Bibliography

A. Askarzadeh. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers and Structures*, 169:1–12, 2016. doi: 10.1016/j.compstruc.2016.03.001. [p443]

E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. [p444]

C. Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000. doi: 10.1016/S0166-3615(99)00046-9. [p443]

A. Gandomi, X. Yang, A. Alavi, and S. Talatahari. Bat algorithm for constrained optimization tasks. *Neural Computing and Applications*, 22:1239–1255, 2013. [p443]

R. B. Gramacy. laGP: Large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46, 2016. doi: 10.18637/jss.v072.i01. [p444]

R. B. Gramacy. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Chapman & Hall/CRC, first edition, 2020. [p441]

R. B. Gramacy, G. A. Gray, S. L. Digabel, H. K. H. Lee, P. Ranjan, G. Wells, and S. M. Wild. Modeling an augmented Lagrangian for blackbox constrained optimization. *Technometrics*, 58(1):1–11, 2016. [p444]

Q. He and L. Wang. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20(1):89–99, 2007. doi: 10.1016/j.engappai.2006.03.003. [p443]

D. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, 13:455–492, 1998. [p444]

H. K. H. Lee, R. B. Gramacy, C. Linkletter, and G. A. Gray. Optimization subject to hidden constraints via statistical emulation. *Pacific Journal of Optimization*, 7:467–478, 2011. [p444]

K. Lee and Z. Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194 (36–38):3902–3933, 2005. doi: 10.1016/j.cma.2004.09.007. [p443]

B. Letham, B. Karrer, G. Ottoni, and E. Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019. [p444]

G. Li, F. Shuang, P. Zhao, and C. Le. An improved butterfly optimization algorithm for engineering design problems using the cross-entropy method. *Symmetry*, 11(8):1049, 2019. doi: 10.3390/sym11081049. [p443]

D. Lindberg and H. K. H. Lee. Optimization under constraints by applying an asymmetric entropy measure. *Journal of Computational and Graphical Statistics*, 24:379–393, 2015. [p443]

M. McDonald and A. Harbaugh. Programmer's documentation for MODFLOW-96, an update to the U.S. geological survey modular finite difference ground-water flow model. Technical report, Open-File Report 96-486, U.S. Geological Survey, 1996. [p443]

S. Mirjalili, S. Mirjalili, and A. Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 2014. doi: 10.1016/j.advengsoft.2013.12.007. [p443]

S. Mirjalili, A. Gandomi, Z. Mirjalili, S. Saremi, H. Fairs, and S. Mirjalili. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, 114: 163–191, 2017. doi: 10.1016/j.advengsoft.2017.07.002. [p443]

J. Mockus, V. Tiesis, and A. Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978. [p443]

T. Pourmohamad. *CompModels: Pseudo Computer Models for Optimization*, 2020. URL `https://CRAN.R-project.org/package=CompModels`. R package version 0.2.0. [p441]

T. Pourmohamad and H. K. H. Lee. Multivariate stochastic process models for correlated responses of mixed type. *Bayesian Analysis*, 11(3):797–820, 09 2016. doi: 10.1214/15-BA976. URL `https://doi.org/10.1214/15-BA976`. [p443]

T. Pourmohamad and H. K. H. Lee. The statistical filter approach to constrained optimization. *Technometrics*, 62(3):303–312, 2019. doi: 10.1080/00401706.2019.1638304. [p441]

T. Pourmohamad and H. K. H. Lee. Bayesian optimization via barrier functions. Technical report, Deptartment of Statistics, University of California, Santa Cruz, 2020. [p444]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`. [p441]

T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York, NY, 2003. [p441]

M. Schonlau, W. J. Welch, and D. Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, pages 11–25, 1998. [p444]

M. L. Stein. *Interpolation of Spatial Data*. Springer, New York, NY, 1999. [p441]

Appendix: Current Computer Models

Table 2 provides a summary of the current computer models that are available in the **CompModels** package. The package is a mix of real-world physics problems, known mathematical functions, and black-box functions, as well as a mix of constrained or unconstrained optimization problems.

Function	Input Dimension	Optimization Type	No. of Constraints
`bbox1()`	2	Constrained	2
`bbox2()`	2	Unconstrained	–
`bbox3()`	2	Unconstrained	–
`bbox4()`	2	Constrained	1
`bbox5()`	3	Unconstrained	–
`bbox6()`	1	Constrained	2
`bbox7()`	8	Constrained	2
`gram()`	2	Constrained	2
`mtp()`	2	Constrained	2
`pressure()`	4	Constrained	4
`sprinkler()`	8	Unconstrained	–
`tension()`	3	Constrained	4

Table 2: Current computer models that are implemented in the **CompModels** package.

Tony Pourmohamad
Genentech, Inc.
1 DNA Way
South San Francisco, CA 94080
United States
`tpourmohamad@gmail.com`

spfilteR: An R package for Semiparametric Spatial Filtering with Eigenvectors in (Generalized) Linear Models

by Sebastian Juhl

Abstract Eigenvector-based Spatial filtering constitutes a highly flexible semiparametric approach to account for spatial autocorrelation in a regression framework. It combines judiciously selected eigenvectors from a transformed connectivity matrix to construct a synthetic spatial filter and remove spatial patterns from model residuals. This article introduces the **spfilteR** package that provides several useful and flexible tools to estimate spatially filtered linear and generalized linear models in R. While the package features functions to identify relevant eigenvectors based on different selection criteria in an unsupervised fashion, it also helps users to perform supervised spatial filtering and to select eigenvectors based on alternative user-defined criteria. Besides a brief discussion of the eigenvector-based spatial filtering approach, this article presents the main functions of the package and illustrates their usage. Comparison to alternative implementations in other R packages highlights the added value of the **spfilteR** package.

Introduction

The presence of spatial autocorrelation in regression residuals constitutes a severe problem in standard inferential statistics as it causes common econometric methods to produce inefficient or even biased and inconsistent parameter estimates (Darmofal, 2015; Goodchild, 2009; Franzese and Hays, 2007). Besides parametric spatial regression techniques, which became the dominant approach to this challenge in the social sciences, spatial filtering techniques offer an alternative approach to handle spatially clustered data. The particular appeal of these alternative semiparametric approaches to spatial autocorrelation arise from their flexibility and the relative ease of estimation and interpretation (e.g., Tiefelsdorf and Griffith, 2007; Getis and Griffith, 2002). Especially the eigenvector-based spatial filtering (ESF) approach pioneered by Griffith (2003, 2000, 1996) has proven to be useful in various academic disciplines.

This article introduces the **spfilteR** package that provides a set of flexible and useful functions to implement the ESF approach in regression models. Besides tools to detect spatial autocorrelation in individual variables and regression residuals by means of the Moran coefficient (MC) (Cliff and Ord, 1981, 1972), the package features easily customizable functions which allow users to perform supervised and unsupervised spatial filtering with eigenvectors. While other R packages like **spatialreg** (Bivand and Piras, 2015) and **spmoran** (Murakami, 2020) also contain implementations of the unsupervised ESF approach, they are less flexible in the specification of eigenvector selection criteria which constitutes the crucial step in the ESF approach. These packages also offer few functions for the supervised selection of eigenvectors.

In contrast, the **spfilteR** package allows users to obtain eigenvectors from a transformed connectivity matrix and to identify a suitable candidate set in order to perform supervised spatial filtering. Alternatively, unsupervised eigenvector selection procedures for different (generalized) linear models based on a stepwise regression procedure are implemented as well. These functions select eigenvectors based on either i) the maximization of model fit, ii) minimization of residual autocorrelation, iii) the statistical significance of residual autocorrelation, or iv) the statistical significance of the candidate eigenvectors. Parameter estimates are obtained by means of ordinary least squares (OLS) for linear models and maximum likelihood estimation (MLE) for generalized linear models (GLMs). The `print`, `summary`, and `plot` methods further facilitate the interpretation and visualization of the results.

After a theoretical description of the ESF approach in a regression framework, this article presents some stylized R code to demonstrate the implementation of the ESF approach using the functions and the synthetic dataset accompanying the **spfilteR** package. It also briefly compares the unsupervised ESF procedures contained in this package to alternative implementations in other R packages. The last section summarizes and concludes this article.

Spatial filtering with eigenvectors

Intuitively, the ESF approach put forth by Griffith (2003, 2000, 1996) and also Tiefelsdorf and Griffith (2007) addresses the problem of spatially autocorrelated regression residuals by partitioning the error term into a spatially structured and a random component (see also Griffith and Chun, 2014). Consider a stylized linear regression model of the following form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e}, \tag{1}$$

where $\boldsymbol{X}$ denotes the matrix of covariates (plus a vector of ones for the intercept term) and $\boldsymbol{\beta}$ is the corresponding parameter vector. If the errors $\boldsymbol{e}$ are not independent but exhibit a non-random spatial pattern, the ESF approach removes this pattern from the disturbances and thereby "whitens" the residuals.

To this end, synthetic proxy variables are generated that reflect the spatial pattern present in model residuals as closely as possible. Subsequently including these synthetic variables as control variables in the regression's mean equation removes the problematic spatial structure from the disturbances and allows the use of standard procedures — such as OLS or MLE — for parameter estimation. Generating these proxy variables that act as the spatial filter requires the decomposition of the transformed and exogenously defined connectivity matrix which represents the dependence structure among the units of analysis.

Eigenfunction decomposition

The eigenfunction (or spectral) decomposition of a transformed connectivity matrix constitutes the core element of the ESF approach. More formally, the decomposition yields

$$\boldsymbol{MVM} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}', \tag{2}$$

where $\boldsymbol{M}$ is a symmetric and idempotent projection matrix, and $\boldsymbol{V}$ is the exogenously specified connectivity matrix which is symmetrized by $\frac{1}{2}(\boldsymbol{W} + \boldsymbol{W}')$. The columns of matrix $\boldsymbol{E}$ are the n mutually uncorrelated eigenvectors obtained from $\boldsymbol{MVM}$, and $\boldsymbol{\Lambda}$ is a diagonal matrix with the corresponding eigenvalues $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ on its main diagonal. Tiefelsdorf and Boots (1995) show that each eigenvector in $\boldsymbol{E}$ represents a distinct map pattern permitted by the units' spatial arrangement and is associated with a certain level of spatial autocorrelation.[1]

The projection matrix is given by $\boldsymbol{M} = \boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$, where $\boldsymbol{I}$ is the identity matrix, and the eigenvectors in $\boldsymbol{E}$ are mutually uncorrelated and orthogonal to the covariates in the design matrix $\boldsymbol{X}$.[2] If only the intercept is included in the construction of the projection matrix, this equation simplifies to $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{1}\boldsymbol{1}'/n)$, where $\boldsymbol{1}$ is an $n \times 1$-dimensional vector of ones. As Tiefelsdorf and Griffith (2007) show, the underlying spatial process generating the data determines both the form of the spatial misspecification in a naïve nonspatial regression and the appropriate specification of $\boldsymbol{M}$.

However, since the number of eigenvectors equals the number of observations in the data, only a subset of eigenvectors can be included in the regression equation.

Eigenvector selection and the spatial filter

Identifying and selecting relevant eigenvectors is decisive in the ESF approach and involves two steps. In a first step, a set of candidate eigenvectors, the search set $\boldsymbol{E}^C \subset \boldsymbol{E}$, needs to be determined based on different criteria. If the model residuals exhibit positive levels of spatial autocorrelation, eigenvectors depicting negative autocorrelation can be discarded since simultaneously including eigenvectors associated with positive and negative spatial autocorrelation can cause problems (Tiefelsdorf and Griffith, 2007). Moreover, eigenvectors portraying negligible levels of spatial autocorrelation can be eliminated as well since they contribute little to the spatial pattern present in model residuals (Chun and Griffith, 2014).

Griffith (2003), for example, proposes a qualitative threshold determining the candidate set by computing MC_i/MC_{max} for all eigenvectors $i \in \{1, 2, \ldots, n\}$, where MC_{max} denotes either the largest positive or the largest negative Moran coefficient of all eigenvectors in $\boldsymbol{E}$. According to this approach, eigenvectors for which $MC_i/MC_{max} \geq 0.25$ should be included in the candidate set $\boldsymbol{E}^C$. Alternatively, Chun et al. (2016) propose a nonlinear function to calculate the ideal size of the candidate set for a given level of spatial autocorrelation and the total number of positive eigenvectors. However, this approach is only applicable if the residuals exhibit positive levels of spatial autocorrelation.

[1]The Moran coefficient for each eigenvector in $\boldsymbol{E}$ can be computed by $MC_i = \lambda_i n / \boldsymbol{1}'\boldsymbol{V}\boldsymbol{1}$ (e.g., Griffith, 1996; Tiefelsdorf and Boots, 1995).

[2]It is important to note that, just like weights in linear models, the presence of a link function corrupts the uncorrelatedness of the eigenvectors in generalized linear models (e.g., Griffith, 2003, 104-105). To check for problematic levels of multicollinearity among the eigenvectors, the function `glmFilter()` in the **spfilteR** package reports the condition number (see also Griffith and Amrhein, 1997).

Once a feasible candidate set is identified, the importance of each eigenvector in $\boldsymbol{E}^C$ needs to be established in a second step. This is typically done by a stepwise regression procedure which sequentially evaluates each eigenvector in the candidate set. To this end, the search algorithm utilizes an objective function in order to determine which eigenvectors to select. The selected eigenvectors $\boldsymbol{E}^* \in \boldsymbol{E}^C$ are the synthetic covariates constituting the spatial filter. Selection criteria commonly employed in the literature include the maximization of model fit statistics (e.g., Tiefelsdorf and Griffith, 2007; Griffith, 2003), the statistical significance of the eigenvectors (e.g., Griffith and Chun, 2014; Le Gallo and Páez, 2013), the minimization of residual autocorrelation (e.g., Tiefelsdorf and Griffith, 2007), or arbitrary combinations of different selection criteria (e.g., Páez, 2019). The aim is to specify an objective function that provides a parsimonious subset of eigenvectors. *Parsimony* here means that $\boldsymbol{E}^*$ minimizes residual autocorrelation with respect to the pre-specified connectivity structure of the filtered model by selecting the smallest number of eigenvectors possible to obtain spatially independent errors (Tiefelsdorf and Griffith, 2007).

Once $\boldsymbol{E}^*$ is established, it can be added to the regression model in Equation (1):

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \overbrace{\underbrace{\boldsymbol{E}^*\boldsymbol{\gamma}}_{filter} + \underbrace{\boldsymbol{\epsilon}}_{noise}}^{\boldsymbol{e}}. \tag{3}$$

Equation (3) depicts the spatially filtered regression model and illustrates how the ESF approach partitions the regression residuals $\boldsymbol{e}$ from Equation (1) into a spatial trend component ($\boldsymbol{E}^*\boldsymbol{\gamma}$) and a random component ($\boldsymbol{\epsilon}$). The selected eigenvectors $\boldsymbol{E}^*$, in conjunction with their parameter estimates $\boldsymbol{\gamma}$, represent the spatial pattern latent in $\boldsymbol{e}$. This term constitutes the synthetic spatial filter that shifts the spatial pattern from the error term to the regression's systematic part. Thereby, it removes the spatial structure from the error term, leaving white noise residuals $\boldsymbol{\epsilon}$.

This stylized filtering scheme directly extends to GLMs, although the link function might corrupt the uncorrelatedness of the eigenvectors. If a substantial amount of multicollinearity among the eigenvectors is present, each eigenvector included in the subset of $\boldsymbol{E}^*$ should be reevaluated whenever a new eigenvector is selected (e.g., Griffith et al., 2019).

The spfilteR package

The stable release version of the **spfilteR** package can be obtained from CRAN.[3] Alternatively, the latest development version is available on GitHub:

```
# install package from CRAN
R> install.packages("spfilteR")

# OR: install development version from GitHub
R> library(devtools)
R> devtools::install_github("sjuhl/spfilteR")
```

Alongside a collection of functions, the package also provides an artificial dataset and a stylized binary connectivity matrix based on the rook scheme of adjacency that connects $n = 100$ units on a regular 10×10 grid. I use this made-up dataset to illustrate key features of the package and its functionality.

To this end, consider a simple linear regression model with a single regressor. Once the model is fitted, the function `MI.resid()` performs a test of residual spatial autocorrelation based on the Moran coefficient (Cliff and Ord, 1981).

```
# load package and data
R> library(spfilteR)
R> data("fakedata")

R> y <- fakedataset$x1
R> X <- fakedataset$x2

R> resid <- resid(lm(y~X))
R> MI.resid(resid,x=X,W=W,alternative="greater")
```

[3]This article is based on version 1.0.0 of the **spfilteR** package.

```
        I         EI       VarI       zI           pI
0.350568 -0.0119261 0.01207299 3.299085 0.0004850019 ***
```

The results suggest that the residuals are spatially autocorrelated, which violates the Gauss-Markov assumption of uncorrelated errors since $cov(\epsilon_i, \epsilon_j) \neq 0 \forall i \neq j$. To address this problem, the **spfilteR** package allows users to implement the ESF approach and to select relevant eigenvectors using different supervised or unsupervised selection procedures.

Supervised spatial filtering

As shown above, the ESF approach starts with the eigenfunction decomposition of a transformed and symmetrized connectivity matrix as depicted in Equation (2). The function `getEVs()` allows users to easily obtain these eigenvectors. Moreover, users have the option to specify covariates that are used in order to construct the projection matrix $\boldsymbol{M}$ via the input `covars`.

```
R> EVs <- getEVs(W=W,covars=NULL)
R> E <- EVs$vectors
```

In addition to the eigenvectors and their corresponding eigenvalues, `getEVs()` also reports the value of the MC associated with each of the eigenvectors.[4] The first eigenvector depicts the spatial pattern permitted by $\boldsymbol{W}$ with the largest possible degree of positive spatial autocorrelation. The second eigenvector displays the pattern associated with the second largest possible degree of positive autocorrelation that is uncorrelated with the first pattern, and so on (Griffith, 1996). Consequently, while the first eigenvectors represent global patterns of positive spatial autocorrelation, the pattern becomes more local as the degree of spatial autocorrelation approaches zero. The last eigenvectors in the set capture patterns of negative autocorrelation (see Figure 1).

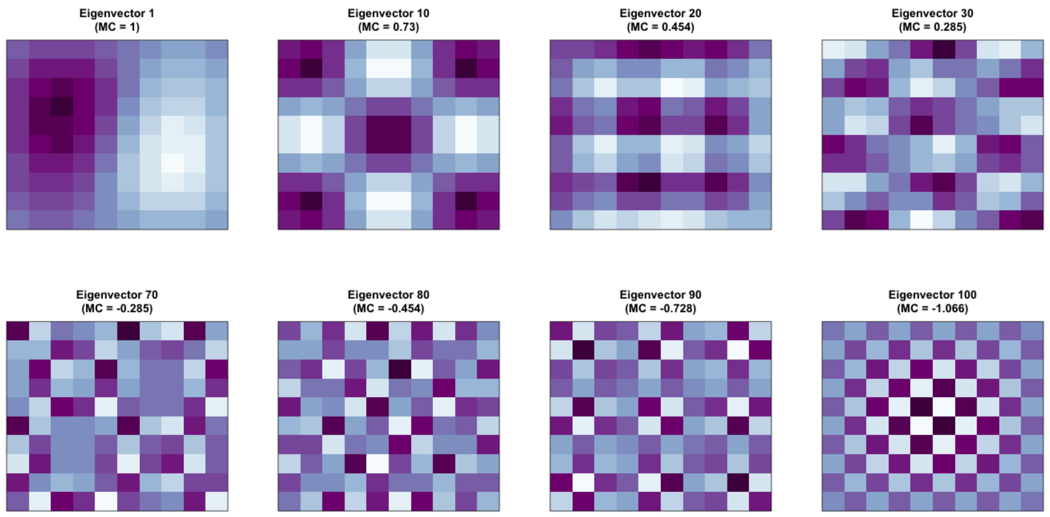


Figure 1: Visualization of Eigenvectors and their respective Moran coefficient (MC). Positive spatial patterns are shown in the first row while negative patterns are depicted in the second row.

Based on the MC values, users can define the candidate set $\boldsymbol{E}^C$ and select relevant eigenvectors based on any desired selection criterium. Using the threshold suggested by Griffith (2003) outlined above, the set $\boldsymbol{E}^C$ consists of 31 eigenvectors. For illustrative purposes, I skip the second step of the eigenvector selection procedure and include all eigenvectors in the ESF model so that $\boldsymbol{E}^C = \boldsymbol{E}^*$.

```
# identify candidate set
R> Ec <- EVs$moran/max(EVs$moran)>=.25

# obtain ESF residuals
R> esf.resid <- resid(lm(y~X+E[,Ec]))

# check for remaining spatial autocorrelation in model residuals
R> MI.resid(esf.resid,x=X,W=W,alternative="greater")
```

[4]To this end, `getEVs()` calls the helper function `MI.ev()`, which calculates the MC for each supplied eigenvector (see also Griffith, 1996; Tiefelsdorf and Boots, 1995).

```
         I         EI       VarI        zI       pI
-0.1836998 -0.0119261 0.01207299 -1.563326 0.941012
```

The results indicate that the ESF approach successfully removed positive spatial autocorrelation from regression residuals. Furthermore, the functions `partialR2()` and `vif.ev()` included in the **spfilteR** package allow users to investigate the proportion of explained variance by each eigenvector and identify potential problems of variance inflation. In this example, eigenvector 13 accounts for about 23.21% of the variance in $\boldsymbol{y}$. Moreover, none of the eigenvectors induces problematic levels of multicollinearity as the variance inflation factor (VIF) of each eigenvectors remains close to 1.

```
R> round(partialR2(y=y,x=X,evecs=E[,Ec]),6)

0.000377 0.060584 0.001004 0.028734 0.020554 0.004804 0.000091 0.007010
0.030418 0.079015 0.004550 0.000012 0.232083 0.011407 0.000959 0.004993
0.001714 0.000094 0.036713 0.044113 0.006588 0.005762 0.001845 0.009648
0.002761 0.031923 0.007490 0.000075 0.004271 0.004042 0.004060

R> vif.ev(x=X,evecs=E[,Ec],na.rm=TRUE)

1.004420 1.001660 1.050409 1.049729 1.011899 1.001588 1.008393 1.000929
1.034209 1.013360 1.000230 1.000027 1.005781 1.022793 1.073397 1.015425
1.014602 1.014900 1.000798 1.002998 1.004616 1.019448 1.001397 1.015900
1.005540 1.000474 1.018344 1.008363 1.000284 1.009756 1.086114
```

Unsupervised spatial filtering

Besides the supervised eigenvector selection procedure, the function `lmFilter()` performs unsupervised spatial filtering and provides parameter estimates by means of OLS. Importantly, users can specify different selection criteria. Thereby, this function eases the implementation of the ESF approach while simultaneously providing considerable flexibility regarding the stepwise selection of eigenvectors. Specifically, the following input arguments allow users to customize the selection procedure and ensure the function's flexibility:

- `objfn` allows users to determine the objective function of the search algorithm determining $\boldsymbol{E}^*$. It supports eigenvector selection based on the adjusted R^2 (`'R2'`), residual spatial autocorrelation (`'MI'`), the significance of eigenvectors (`'p'`), and the significance level of residual spatial autocorrelation (`'pMI'`). Alternatively, all eigenvectors may be included by spefifying `objfn='all'`, implying that no selection takes place.
- `MX` (optional) specifies the covariates used to construct the projection matrix $\boldsymbol{M}$. As Tiefelsdorf and Griffith (2007) show, the specification of $\boldsymbol{M}$ is directly linked to the form of the spatial misspecification in the unfiltered naïve regression model.
- `sig` and `bonferroni` indicate the significance level if the search algorithm selects eigenvectors based on their significance or the significance of residual spatial autocorrelation. If `bonferroni=TRUE` and `objfn='p'`, the significance level will be adjusted in order to account for inflated Type-I errors. If `objfn='pMI'`, `bonferroni` is automatically set to `FALSE`.
- `positive` (`TRUE` or `FALSE`) restricts the eigenvector search to those eigenvectors associated with positive levels of spatial autocorrelation.
- `ideal.setsize` (`TRUE` or `FALSE`) determines the ideal size of the candidate set $\boldsymbol{E}^C$ according to the formula given in Chun et al. (2016). Note that this is only valid when filtering for positive spatial autocorrelation.
- `alpha` allows users to specify a threshold for the inclusion of eigenvectors in the candidate set based on their MC values (see Griffith, 2003).
- `tol` sets a tolerance threshold for remaining residual autocorrelation if `objfn='MI'`. Once the level of residual autocorrelation reaches the threshold, the selection procedure terminates.
- `boot.MI` (optional) takes integers indicating the number of bootstrap permutations in order to estimate the variance of the Moran test for residual autocorrelation.

These arguments allow users to customize the ESF model and obtain parameter estimates by using a single function call and only a few lines of code. While the `print` method for the output — an object of class `"spfilter"` — only reports the number of selected eigenvectors in $\boldsymbol{E}^*$ and the size of the candidate set $\boldsymbol{E}^C$, the `summary` method provides a host of useful additional information.

```
R> (esf <- lmFilter(y=y,x=X,W=W,objfn="p",sig=.1,bonferroni=TRUE
+                     ,positive=TRUE,ideal.setsize=TRUE))

3 out of 22 candidate eigenvectors selected

R> summary(esf,EV=TRUE)

        - Spatial Filtering with Eigenvectors (Linear Model)  -

Coefficients (OLS):
            Estimate        SE      p-value
(Intercept) 9.370881 0.71253832 4.103548e-23 ***
beta_1      0.975771 0.08536198 1.511830e-19 ***

Adjusted R-squared:
  Initial  Filtered
0.4673945 0.6534442

Filtered for positive spatial autocorrelation
3 out of 22 candidate eigenvectors selected
Objective Function: "p" (significance level=0.1)
Bonferroni correction: TRUE (adjusted significance level=0.00455)

Summary of selected eigenvectors:
       Estimate       SE      p-value  partialR2      VIF        MI
ev_13 -9.552977 1.626696 6.290028e-08 0.23208263 1.005781 0.6302019 ***
ev_10 -5.571465 1.632824 9.483754e-04 0.07901543 1.013360 0.7303271 ***
ev_2   4.900028 1.623316 3.261057e-03 0.06058390 1.001660 1.0004147  **

Moran's I ( Residuals):
           Observed    Expected   Variance        z      p-value
Initial  0.3505680 -0.01192610 0.01207299 3.299085 0.0004850019 ***
Filtered 0.1397003 -0.03703186 0.02417938 1.136562 0.1278607838
```

Besides the parameter estimates of the filtered model, the summary method provides information on the fit of the filtered and the unfiltered models, the objective function, and the Moran test for residual autocorrelation. If users specify EV=TRUE, information on the included eigenvectors in the order of their selection will be displayed as well. Just like above, we see that the eigenvector 13, for example, explains 23.21% of the variance, and the VIF indicates no problems of multicollinearity in the filtered model. The adjusted R^2 also shows that the ESF approach considerably improves model fit.

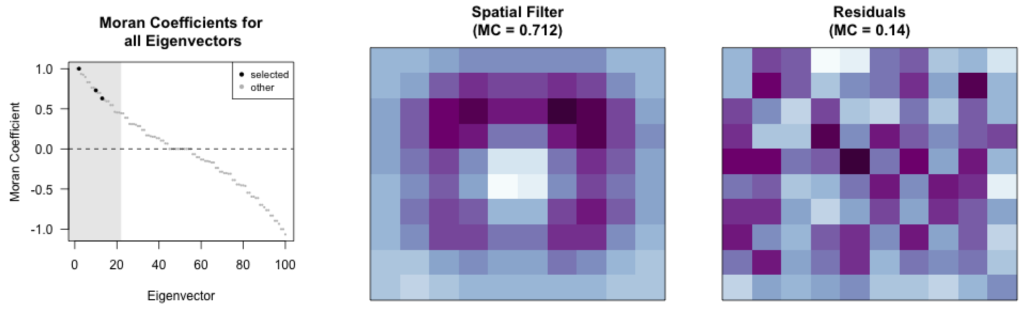


Figure 2: Plotting method for objects of class "spfilter" (left), spatial pattern captured by the filter and calculated by MI.sf() (center), and spatial patterns of filtered residuals (right).

Finally, the left part of Figure 2 demonstrates the plotting method for objects of class "spfilter" which is produced by plot(esf). It visualizes the MC of each eigenvector and highlights the ones selected by the unsupervised selection procedure. The grey shaded area illustrates the candidate set $\boldsymbol{E}^C$ from which the eigenvectors in $\boldsymbol{E}^*$ are selected. Figure 2 further depicts the spatial pattern of the spatial filter (center) and the filtered residuals (right). The function MI.sf() computes the MC value associated with the map pattern depicted by the spatial filter $\boldsymbol{E}^*\gamma$ in Equation (3) (e.g., Le Gallo and Páez, 2013).

Spatial filtering in generalized linear models

The ESF methodology directly extends to GLMs. In fact, one of the advantages of the filtering approach as compared to parametric spatial regression models in this context is that parameter estimates can be obtained by standard MLE and do not require the application of more sophisticated estimation techniques (Griffith et al., 2019).

Besides the supervised filtering procedure, the function `glmFilter()` from the **spfilteR** package allows users to perform unsupervised spatial filtering in GLMs. While its usage is purposefully similar to the function `lmFilter()` introduced above, GLMs require some adjustments of the filtering procedure. As a result, `glmFilter()` not only uses MLE instead of OLS to obtain parameter estimates but also differs in some of the function's input. Hence, in addition the input already discussed above, `glmFilter()` differs to `lmFilter()` with respect to the following input arguments:

- `objfn` defines the eigenvector selection criterium. Possible criteria are the maximization of model fit (`'AIC'` or `'BIC'`), minimization of residual autocorrelation (`'MI'`), the significance level of candidate eigenvectors (`'p'`), the significance of residual spatial autocorrelation (`'pMI'`), or all eigenvectors in the candidate set (`'all'`).
- `model` specifies the type of model to be estimated. The current version of **spfilteR** (version 1.0.0) supports `'probit'`, `'logit'`, and `'poisson'` as input.
- `optim.method` determines the method used to optimize the likelihood function.
- `min.reduction` takes values in the interval $[0, 1)$. It defines the minimum level of reduction in the AIC or BIC (if either selection criterium is chosen) relative to the current AIC/ BIC a candidate eigenvector needs to achieve in order to be included in the spatial filter.
- `resid.type` allows users to specify the type of residuals which is used to calculate the MC value. Valid arguments are `'raw'`, `'deviance'`, and the default option `'pearson'`.

Implementing the ESF approach in GLMs using `glmFilter()` requires as few lines of code as using the `lmFilter()` function in the context of linear regression models. The following example demonstrates the ease of implementation in the context of a logit, a probit, and a Poisson regression model:

```
# define DVs
R> y.bin <- fakedataset$indicator
R> y.count <- fakedataset$count

# seed (because of 'boot.MI')
set.seed(123)

# logit model
R> (esf.logit <- glmFilter(y=y.bin,x=NULL,W=W,objfn="p",model="logit",optim.method="BFGS"
+                           ,sig=.05,bonferroni=FALSE,resid.type="pearson",boot.MI=100))

3 out of 31 candidate eigenvectors selected

# probit model
R> (esf.probit <- glmFilter(y=y.bin,x=NULL,W=W,objfn="BIC",model="probit"
+                            ,optim.method="BFGS",min.reduction=0,resid.type="deviance"
+                            ,boot.MI=100))

2 out of 31 candidate eigenvectors selected

# poisson model
R> (esf.poisson <- glmFilter(y=y.count,x=NULL,W=W,objfn="pMI",model="poisson"
+                             ,optim.method="BFGS",sig=.1,resid.type="pearson"
+                             ,boot.MI=100))

0 out of 31 candidate eigenvectors selected
```

Of course, users can also define their own eigenvector selection criteria or apply the ESF approach to models currently not supported by the `glmFilter()` function. Just like for linear regression models illustrated above, the function `getEVs()` performs the eigenfunction decomposition of the transformed and symmetrized connectivity matrix, and users can implement a supervised selection procedure using the standard `glm()` function.

A brief comparison to other R packages

Of course, alternative implementations of the ESF approach outlined here exist in other R packages as well. While these packages are highly useful for spatial analysts, the **spfilteR** package offers a couple of notable extensions that improve these existing implementations.[5]

The **spmoran** package developed by Murakami (2020) contains different functions for estimating eigenvector-based spatial additive mixed models. Although the function `esf()` estimates a linear spatial filtering model, the main advantages of this package are the estimation of the random effects ESF model (e.g., Murakami and Griffith, 2019) and the fast approximation of the eigenfunction decomposition, which makes this package especially useful for large datasets. Moreover, users can also use the functions `meigen()` and `meigen_f()` to obtain eigenfunctions and perform supervised eigenvector selection.

At the same time, the eigenvector selection criteria implemented in `esf()` only allow for the identification of relevant eigenvectors based on model fit statistics such as the adjusted R^2, the AIC, or the BIC. The specification of the projection matrix $\boldsymbol{M}$ also does not allow for the inclusion of covariates. Furthermore, the **spmoran** package does not support the ESF approach in the context of GLMs.

Alternatively, the **spatialreg** package, which encompasses a great variety of different spatial estimation techniques, not only provides the `SpatialFiltering()` function estimating spatially filtered linear models. It also allows for the estimation of spatially filtered GLMs by using `ME()`. Yet, both of these functions utilize an objective function that selects eigenvectors based on the overall reduction of residual autocorrelation. While it is possible to restrict the candidate set size and to customize the level of remaining autocorrelation at which the search terminates, users cannot select alternative objective functions. Moreover, `ME()` does not allow for the inclusion of covariates in the construction of $\boldsymbol{M}$. Since there is no function to perform the eigenfunction decomposition shown in Equation (2), the package offers no support for supervised spatial filtering.

Therefore, the **spfilteR** package provides additional flexibility – especially for the estimation of filtered linear and generalized linear models where the ESF approach is predominantly applied. Since the eigenvector selection procedure is the crucial step in the ESF approach, the options provided by `lmFilter()` and `glmFilter()` allow users to tailor the ESF procedure to their specific needs. The option to estimate the ideal size of the eigenvector candidate set $\boldsymbol{E}^C$ according to Chun et al. (2016), the specification of different residual types in GLMs, and the ability to define a threshold for the minimum increase in model fit when an objective function is chosen accordingly are examples of features unique to the **spfilteR** package.

Despite this additional flexibility, the functions that perform unsupervised eigenvector selection are very easy to use and only require a minimum of code. Moreover, the `getEVs()` command and several additional helper functions such as `MI.ev()`, `MI.sf()`, `partialR2()`, and `vif.ev()` introduced above facilitate the estimation of spatially filtered (generalized) linear models. Consequently, while the **spmoran** and the **spatialreg** packages cover additional model types and estimation strategies, the flexibility provided by the **spfilteR** package constitutes a great advantage in the most common applications of the ESF approach.

Summary

This article briefly covers the basics of spatial filtering with eigenvectors and introduces the **spfilteR** package. Using the synthetic dataset provided by the package, it discusses the main functions and their implementation in the context of supervised and unsupervised spatial filtering as well as its extension to GLMs. By comparing the package to alternative implementations of the ESF approach, this article highlights that the flexibility provided by the **spfilteR** package constitutes an important improvement in settings where the ESF approach is commonly applied.

Acknowledgments

Funding was provided by the German Research Foundation (DFG) through the Collaborative Research Center (SFB) 884 (grant number: 139943784). This work was also supported by a postdoc fellowship of the German Academic Exchange Service (DAAD).

[5]Note that other R packages such as **adespatial** (Dray et al., 2020) and **vegan** (Oksanen et al., 2019) also provide tools to eigendecompose a transformed connectivity matrix and implement the supervised ESF approach. Yet, since these packages are not primarily concerned with ESF and do not support unsupervised spatial filtering, I do not discuss them in greater detail here.

Bibliography

R. Bivand and G. Piras. Comparing implementations of estimation methods for spatial econometrics. *Journal of Statistical Software*, 63(18):1–36, 2015. URL https://www.jstatsoft.org/v63/i18/. [p450]

Y. Chun and D. A. Griffith. A quality assessment of eigenvector spatial filtering based parameter estimates for the normal probability model. *Spatial Statistics*, 10:1–11, 2014. URL https://doi.org/10.1016/j.spasta.2014.04.001. [p451]

Y. Chun, D. A. Griffith, M. Lee, and P. Sinha. Eigenvector selection with stepwise regression techniques to construct eigenvector spatial filters. *Journal of Geographical Systems*, 18(1):67–85, 2016. URL https://doi.org/10.1007/s10109-015-0225-3. [p451, 454, 457]

A. D. Cliff and J. K. Ord. Testing for spatial autocorrelation among regression residuals. *Geographical Analysis*, 4(3):267–284, 1972. URL https://doi.org/10.1111/j.1538-4632.1972.tb00475.x. [p450]

A. D. Cliff and J. K. Ord. *Spatial Processes: Models & Applications*. Advances in Spatial Science. Pion, London, 1981. ISBN 9780850860818. [p450, 452]

D. Darmofal. *Spatial Analysis for the Social Sciences*. Analytical Methods for Social Research. Cambridge University Press, New York, USA, 2015. URL https://doi.org/10.1017/CBO9781139051293. [p450]

S. Dray, D. Bauman, G. Blanchet, D. Borcard, S. Clappe, G. Guenard, T. Jombart, G. Larocque, P. Legendre, N. Madi, and H. H. Wagner. *adespatial: Multivariate Multiscale Spatial Analysis*, 2020. URL https://CRAN.R-project.org/package=adespatial. R package version 0.3-8. [p457]

R. J. Franzese and J. C. Hays. Spatial econometric models of cross-sectional interdependence in political science panel and time-series-cross-section data. *Political Analysis*, 15(2):140–164, 2007. URL https://doi.org/10.1093/pan/mpm005. [p450]

A. Getis and D. A. Griffith. Comparative spatial filtering in regression analysis. *Geographical Analysis*, 34(2):130–140, 2002. URL https://doi.org/10.1111/j.1538-4632.2002.tb01080.x. [p450]

M. F. Goodchild. What problem? spatial autocorrelation and geographic information science. *Geographical Analysis*, 41(4):411–417, 2009. URL https://doi.org/10.1111/j.1538-4632.2009.00769.x. [p450]

D. Griffith and C. G. Amrhein. *Multivariate Statistical Analysis for Geographers*. Prentice Hall, Englewood Cliffs, 1997. ISBN 9780136056928. [p451]

D. Griffith and Y. Chun. Spatial autocorrelation and spatial filtering. In M. M. Fischer and P. Nijkamp, editors, *Handbook of Regional Science*, pages 1477–1507. Springer, Berlin, Heidelberg, 2014. URL https://doi.org/10.1007/978-3-642-23430-9_72. [p451, 452]

D. A. Griffith. Spatial autocorrelation and eigenfunctions of the geographic weights matrix accompanying geo-referenced data. *The Canadian Geographer*, 40(4):351–367, 1996. URL https://doi.org/10.1111/j.1541-0064.1996.tb00462.x. [p450, 451, 453]

D. A. Griffith. A linear regression solution to the spatial autocorrelation problem. *Journal of Geographical Systems*, 2(2):141–156, 2000. URL https://doi.org/10.1007/PL00011451. [p450]

D. A. Griffith. *Spatial Autocorrelation and Spatial Filtering: Gaining Understanding Through Theory and Scientific Visualization*. Advances in Spatial Science. Springer, Berlin, Heidelberg, 2003. URL https://doi.org/10.1007/978-3-540-24806-4. [p450, 451, 452, 453, 454]

D. A. Griffith, Y. Chun, and B. Li. *Spatial Regression Analysis Using Eigenvector Spatial Filtering*. Spatial Econometrics and Spatial Statistics. Elsevier, London, 2019. ISBN 9780128150436. URL https://doi.org/10.1016/B978-0-12-815043-6.09990-0. [p452, 456]

J. Le Gallo and A. Páez. Using synthetic variables in instrumental variable estimation of spatial series models. *Environment and Planning A: Economy and Space*, 45(9):2227–2242, 2013. URL https://doi.org/10.1068/a45443. [p452, 455]

D. Murakami. *spmoran: Moran Eigenvector-Based Scalable Spatial Additive Mixed Modeling*, 2020. URL https://CRAN.R-project.org/package=spmoran. R package version 0.2.0-2. [p450, 457]

D. Murakami and D. A. Griffith. Eigenvector spatial filtering for large data sets: Fixed and random effects approaches. *Geographical Analysis*, 51(1):23–49, 2019. URL https://doi.org/10.1111/gean.12156. [p457]

J. Oksanen, F. G. Blanchet, M. Friendly, R. Kindt, P. Legendre, D. McGlinn, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, E. Szoecs, and H. H. Wagner. *vegan: Community Ecology Package*, 2019. URL https://CRAN.R-project.org/package=vegan. R package version 2.5-6. [p457]

A. Páez. Using spatial filters and exploratory data analysis to enhance regression models of spatial data. *Geographical Analysis*, 51(3):314–338, 2019. URL https://doi.org/10.1111/gean.12180. [p452]

M. Tiefelsdorf and B. Boots. The exact distribution of moran's i. *Environment and Planning A: Economy and Space*, 27(6):985–999, 1995. URL https://doi.org/10.1068/a270985. [p451, 453]

M. Tiefelsdorf and D. A. Griffith. Semiparametric filtering of spatial autocorrelation: The eigenvector approach. *Environment and Planning A: Economy and Space*, 39(5):1193–1221, 2007. URL https://doi.org/10.1068/a37378. [p450, 451, 452, 454]

Sebastian Juhl
Collaborative Research Center 884
University of Mannheim
B6, 30-32
68159 Mannheim
Germany
ORCiD: https://orcid.org/0000-0002-7123-5398
sebastian.juhl@gess.uni-mannheim.de

SIQR: An R Package for Single-index Quantile Regression

by Tianhai Zu and Yan Yu

Abstract We develop an R package **SIQR** that implements the single-index quantile regression (SIQR) models via an efficient iterative local linear approach in Wu et al. (2010). Single-index quantile regression models are important tools in semiparametric regression to provide a comprehensive view of the conditional distributions of a response variable. It is especially useful when the data is heterogeneous or heavy-tailed. The package provides functions that allow users to fit SIQR models, predict, provide standard errors of the single-index coefficients via bootstrap, and visualize the estimated univariate function. We apply the R package **SIQR** to a well-known Boston Housing data.

Introduction

Single-index quantile regression (Wu et al., 2010) generalizes the seminal work of linear quantile regression of Koenker and Bassett (1978) by projecting the d-dimensional covariate $\mathbf{x}$ to a univariate index $\mathbf{x}\boldsymbol{\beta}$ and allowing a flexible univariate function $g(\mathbf{x}\boldsymbol{\beta})$. Quantile regression is often of great interest, especially when heterogeneity is present. Applications lie in a variety of fields, such as growth curves and reference charts in medicine; survival analysis when a given covariate may have a different effect on individuals with different levels of risks; value at risk calculation and wage and income studies in financial economics; high peak electricity demand in terms of weather characteristics in utility and energy; modeling rainfall, river flow, and air pollution in environmental modeling (see a survey inYu et al. 2003).

Single-index quantile regression (SIQR) is a flexible semiparametric quantile regression model for analyzing heterogeneous data. The SIQR model has some appealing features: (i) It can provide a comprehensive view of the conditional distribution of a response variable given d-dimensional covariates by examining the full spectrum of conditional quantiles. This is especially important for complex heterogeneous data. (ii) The single-index structure is flexible to accommodate nonlinearity while avoiding the curse of dimensionality. It can also implicitly model some interactions among the covariates. Some interesting interpretations of the single-index parameter may be preserved. (iii) The quantile regression approach is robust to heavy-tailed distributions.

We present a package **SIQR** in R that implements the iterative local linear approach to the single-index quantile regression in Wu et al. (2010). The unknown univariate function is estimated by local linear estimation. The key algorithm can be decomposed into two efficient estimation steps on augmented data through local linear approximation and some equivalent formulation of the expected loss. Essentially, it iterates between two linear quantile regressions utilizing the state-of-the-art R package **quantreg**.

We apply our R package, **SIQR**, to the well-known Boston Housing data (1978) that is available in the R default library. The data has a total of 506 observations, and the response variable of interest is the median price of owner-occupied homes on the census tracts in suburban Boston from the 1970 census. The response variable and some covariates are left-skewed. Clearly, quantile regression is a natural tool to analyze the data (e.g., Chaudhuri et al. 1997; Yu and Lu 2004; Wu et al. 2010; Kong and Xia 2012). We organize the rest of the paper as follows. In the next section, we review the SIQR models. Next, we discuss the estimation algorithms implemented in this package. The section following describes the main features of the functions provided. Section "Real Data Analysis and Simulation" illustrates the use of **SIQR** in R for Boston housing data and a simulation study. The last section concludes the paper.

An overview for single-index quantile regression

Data structure and model settings

We develop an R package for the single-index quantile regression for semiparametric estimation with d-dimensional covariates. Let Y be the response variable and $\mathbf{X}$ be the covariate vector. Suppose there are n observations $\left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^n$ of $(\mathbf{X} = \mathbf{x}, Y = y)$. Given $\tau \in (0,1)$ and covariates $\mathbf{x}_i$, the single-index quantile model for the τ-th conditional quantile of the i-th observation is

$$q_\tau(Y = y_i | \mathbf{X} = \mathbf{x}_i) = g_\tau(\mathbf{x}_i \boldsymbol{\beta}_\tau), \tag{1}$$

where y_i is a real valued response, covariate $\mathbf{x}_i$ is a d-dimensional row vector, the single-index parameter $\boldsymbol{\beta}_\tau$ is a column vector in $\mathbf{R}^d$, and the univariate function $g_\tau : \mathbf{R} \to \mathbf{R}$ is subject to different τ. For identifiability, the single index parameter $\|\boldsymbol{\beta}_\tau\| = 1$ and the first non-zero element of $\boldsymbol{\beta}_\tau$ is positive (Yu and Ruppert, 2002). The projection $\mathbf{x}\boldsymbol{\beta}$ is often termed as the "single index". When g_τ is linear, single-index quantile regression model (1) reduces to the seminal work of linear quantile regression of Koenker and Bassett (1978).

Review of local linear estimation for single-index quantile regression

We implement the local linear estimation for single-index quantile regression (1) (Wu et al., 2010). For notational convenience, we omit the subscript τ in g_τ and $\boldsymbol{\beta}_\tau$. The true parameter vector $\boldsymbol{\beta}$ is the minimizer of

$$E\left[\rho_\tau\left(y - g(\mathbf{x}\boldsymbol{\beta})\right)\right], \tag{2}$$

where $\rho_\tau(u) = |u| + (2\tau - 1)u$ is the loss function, often termed as the “check" function in quantile regression. $g(\cdot)$ is the unknown univariate function. Constraint $\|\boldsymbol{\beta}\| = 1, \beta_1 > 0$ is imposed for identifiability. The above expected loss can be equivalently written as

$$E\left\{E\left[\rho_\tau\left(y - g(\mathbf{x}\boldsymbol{\beta})\right)|\mathbf{x}\boldsymbol{\beta}\right]\right\}, \tag{3}$$

where $E\left[\rho_\tau\left(y - g(\mathbf{x}\boldsymbol{\beta})\right)|\mathbf{x}\boldsymbol{\beta}\right]$ is the conditional expected loss and $g(\cdot)$ is the τth conditional quantile given the single-index parameter $\boldsymbol{\beta}$.

We adopt a local linear approximation. In particular, for $\mathbf{x}_i\boldsymbol{\beta}$ "close” to u, we can approximate the τth conditional quantile at $\mathbf{x}_i\boldsymbol{\beta}$ linearly via

$$g(\mathbf{x}_i\boldsymbol{\beta}) \approx g(u) + g'(u)(\mathbf{x}_i\boldsymbol{\beta} - u) = a + b(\mathbf{x}_i\boldsymbol{\beta} - u),$$

where we define $a \equiv g(u)$ and $b \equiv g'(u)$.

Now, we can minimize the sample analogue of (2) below as in Yu and Jones (1998) with respect to (a, b) with local linear estimation

$$\sum_{i=1}^{n} \rho_\tau\left(y_i - a - b(\mathbf{x}_i\boldsymbol{\beta} - u)\right) K\left(\frac{\mathbf{x}_i\boldsymbol{\beta} - u}{h}\right), \tag{4}$$

where $K(\cdot)$ is the kernel function and h is the bandwidth.

We further average (4) over u and obtain the sample analog of (3). The objective function below is used to estimate our single-index quantile regression model (1),

$$\sum_{j=1}^{n}\sum_{i=1}^{n} \rho_\tau\left(y_i - a_j - b_j(\mathbf{x}_i\boldsymbol{\beta} - \mathbf{x}_j\boldsymbol{\beta})\right)\omega_{ij}, \tag{5}$$

where

$$\omega_{ij} = \frac{K_h(\mathbf{x}_i\boldsymbol{\beta} - \mathbf{x}_j\boldsymbol{\beta})}{\sum_{k=1}^{n} K_h(\mathbf{x}_k\boldsymbol{\beta} - \mathbf{x}_j\boldsymbol{\beta})} \tag{6}$$

and $K_h(\cdot) = K(\cdot/h)/h$. We implement minimizing (5) iteratively with a detailed algorithm described next.

Bandwidth is a critical smoothing parameter that tunes the smoothness of the fitted function in local estimation. We implement the choice of the optimal bandwidth h_τ as advocated in Wu et al. (2010) through a computationally-expedient rule-of-thumb:

$$h_\tau = h_m\left\{\tau(1-\tau)/\phi\left(\Phi^{-1}(\tau)\right)^2\right\}^{1/5}, \tag{7}$$

where $\phi(\cdot)$ is the probability density function and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. Here, $h_m = \left\{\frac{[\int K^2(v)dv][var(y|\mathbf{x}\boldsymbol{\beta}=u)]}{n[\int v^2 K(v)dv]^2[\frac{d^2}{du^2}E(y|\mathbf{x}\boldsymbol{\beta}=u)]^2[f_{u_0}(u)]}\right\}^{1/5}$ is the optimal bandwidth in mean regression, which is easily obtainable from many existing packages (Ruppert et al., 1995).

Algorithm

We present the main algorithm for fitting the single-index quantile regression (SIQR) with local linear estimation in detail as following:

```
Input: Quantile level $\tau \in (0,1)$, $d$-dimensional covariate vector $\mathbf{X} = \mathbf{x}$, and a response vector
       $Y = \mathbf{y}$.
Output: The estimated quantile single-index parameter $\widehat{\boldsymbol{\beta}}_\tau$ and fitted conditional quantile
       $\hat{q}_\tau(Y = \mathbf{y}|\mathbf{X} = \mathbf{x})$. The univariate function estimate $\hat{g}_\tau(\cdot)$.
1 Obtain an initial estimate $\hat{\boldsymbol{\beta}}^{(0)}$ of the quantile single-index parameter $\boldsymbol{\beta}$ from a linear quantile
    regression model (default) or a user-provided initial list. Standardize the initial estimate such
    that $||\hat{\boldsymbol{\beta}}^{(0)}|| = 1$ and $\hat{\beta}_1^{(0)} > 0$.
2 Given $\hat{\boldsymbol{\beta}}$, obtain $\{\hat{a}_j, \hat{b}_j\}_{j=1}^n$ by solving a series of the following
      $$\min_{(a_j,b_j)} \sum_{i=1}^n \rho_\tau \left( y_i - a_j - b_j(\mathbf{x}_i - \mathbf{x}_j)\hat{\boldsymbol{\beta}} \right) \omega_{ij}, \quad (8)$$
    where the weights $\omega_{ij}$ is defined in (6). The bandwidth $h$ is chosen optimally following a
    rule-of-the-thumb criterion in (7).
3 Given $\{\hat{a}_j, \hat{b}_j\}_{j=1}^n$, obtain $\hat{\boldsymbol{\beta}}$ by solving
      $$\min_{\boldsymbol{\beta}} \sum_{j=1}^n \sum_{i=1}^n \rho_\tau \left( y_i - \hat{a}_j - \hat{b}_j(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\beta} \right) \omega_{ij}, \quad (9)$$
    with $\omega_{ij}$ evaluated at $\boldsymbol{\beta}$ and $h$ from step 2.
4 Repeat Steps 2 and 3 until convergence.
5 Finally, we estimate $g(\cdot)$ at any $u$ by $\hat{g}(\cdot; h, \hat{\boldsymbol{\beta}}) = \hat{a}$, where
      $$(\hat{a}, \hat{b}) = \arg\min_{(a,b)} \sum_{i=1}^n \rho_\tau \left( y_i - a - b(\mathbf{x}_i\hat{\boldsymbol{\beta}} - u) \right) K_h(\mathbf{x}_i\hat{\boldsymbol{\beta}} - u).$$
    Obtain the final fitted conditional quantile $\hat{q}_\tau(Y = \mathbf{y}|\mathbf{X} = \mathbf{x})$ from model (1).
```

The above algorithm effectively decomposes (5) into two steps that can be achieved by two standard linear quantile regression procedures in Steps 2 and 3. In Step 3, we further note that (9) can be written as

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \sum_{j=1}^n \sum_{i=1}^n \rho_\tau \left(y_i - \hat{a}_j - \hat{b}_j(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\beta} \right) \omega_{ij} \\
&= \arg\min_{\boldsymbol{\beta}} \sum_{j=1}^n \sum_{i=1}^n \rho_\tau \left(y_{ij}^* - \mathbf{x}_{ij}^*\boldsymbol{\beta} \right) \omega_{ij},
\end{aligned}$$

where $y_{ij}^* = y_i - \hat{a}_j$, $\mathbf{x}_{ij}^* = \hat{b}_j(\mathbf{x}_i - \mathbf{x}_j)$, and ω_{ij} evaluated at the previous step, $i, j = 1, \cdots, n$. Given $\hat{a}_j$'s and $\hat{b}_j$'s, we can estimate $\boldsymbol{\beta}$ through usual linear quantile regression without intercept (*regression-through-origin*) on n^2 "observations" $\{y_{ij}^*, \mathbf{x}_{ij}^*\}_{i,j=1}^n$ with known weights $\{\omega_{ij}\}_{i,j=1}^n$ evaluated at the estimate of $\boldsymbol{\beta}$ from the previous iteration.

We can see that (9) is an alternative to (8). Adopting (9) yields some advantages: (i) It uses all the data and is more efficient in estimation; (ii) The double sum in (9) effectively increases the "augmented" sample size to n^2, similar to the minimum average variance estimation (MAVE) in the mean regression (Xia and Härdle, 2006).

The SIQR package

The R package **SIQR** consists of one core estimation function `siqr` and some supporting functions such as visualization tool `plot.siqr` and summary function `summary.siqr`. The R package SIQR depends on the R packages **stats**, **quantreg**, **KernSmooth**.

Main fitting function

The main estimation function `siqr` implements the iterative local linear approach to the single-index quantile regression in Wu et al. (2010).

The usage and input arguments of the main fitting function `siqr` are summarized as follows:

```
siqr(y, X, tau=0.5, h=NULL, beta.initial=NULL, se.method = NULL, maxiter=30, tol=1e-8)
```

This function takes two required arguments: the response variable y in vector format, the covariate matrix X. Please note that all the input covariates are required to be numeric variables.

This function also takes several optional arguments for finer controls. The optional argument `tau` is the quantile index, which specifies the left-tail probability. The default value of `tau` is 0.5, which refers to a single-index median regression. The optional argument `h` is the bandwidth in local linear quantile regression. Users can either provide a bandwidth or let the algorithm decide the optimal bandwidth as advocated in Wu et al. (2010) by setting this argument to NULL as default. The optional argument `beta.initial` is a numeric vector of the same length as the dimensionality of covariates. The users can use this argument to pass in any appropriate user-defined initial single-index coefficients based on prior information or domain knowledge. The default value is NULL, which instructs the function to estimate the initial single-index coefficients by linear quantile regression. The optional argument `se.method` is a character variable that specifies the method to obtain the standard error of estimated single-index coefficients. The default value is NULL to skip the calculation of standard error while the bootstrap-based method is available with "bootstrap". The optional argument `maxiter` and `tol` are control parameters that specify the criteria to terminate the iteration process. Although the algorithm normally converges quickly, the default `maxiter` and `tol` are set to 30 and 1e-8, respectively.

Other functions

We also provide several supporting functions:

```
summary.siqr(siqr.object)
print.summary.siqr(siqr.object)
```

The functions `summary.siqr` and `print.summary.siqr` provide detailed information related to the fitted model and summarize the results as illustrated in the next section. These two functions can be called directly by applying functions print and summary to the siqr.object.

```
plot.siqr(siqr.object, data.points = TRUE, bootstrap.interval=FALSE)
```

This function plots the fitted quantiles against the single-index term from an SIQR-fitted model object. By default, this function will also plot the observed data points in addition to the fitted quantiles to visualize the fitness of the model. One can remove the data points by setting the optional argument `data.points` to FALSE. Pointwise confidence interval will be added to the plot if the optional argument `bootstrap.interval` is set to TRUE.

```
simulation_data <- generate.data(n, true.theta=NULL, sigma=0.1,
setting="setting1", ncopy=1)
```

To help perform simulation studies, the function `generate.data` generates a size n data from two different settings: (i) a sine-bump model; and (ii) a location-scale model as in Wu et al. (2010). Users can define the single-index coefficients $\boldsymbol{\beta}$ via the argument `true.beta` and the noise level via `sigma`. If no `true.beta` was provided, the function will use $(1,1,1)^\top/\sqrt{3}$ for setting 1 and $(1,2)^\top/\sqrt{5}$ for setting 2 as the default. The last optional argument `ncopy` generates multiple copies of data for Monte Carlo simulations.

Real Data and Simulations

Boston Housing data

We consider the Boston housing data to demonstrate the real data application of the proposed R package **SIQR**. This dataset contains the median value of houses (in $1000's), medv, in 506 tracts in Boston and 13 other socio-demographic related variables. This data has been investigated by many studies. Heterogeneity and some non-linear dependence of medv on predictor variables have been found by previous researchers. The dataset is maintained at the StatLib library of Carnegie Mellon University and can be found at the R built-in package **MASS**.

We focus on the following four covariates: RM, the average number of rooms per dwelling; TAX, the full-value property tax (in $) per $10,000; PTRATIO, the pupil-teacher ratio by town; and LSTAT, percentage of the lower status of the population as in Opsomer and Ruppert (1998), Yu and Lu (2004), and Wu et al. (2010). Following previous studies, we take logarithmic transformations on TAX and LSTAT and center the dependent variable medv around zero.

We use the following codes to load data from **MASS** and pre-process as discussed above. We fit a single-index quantile regression with $\tau = 0.25, 0.50, 0.75$ to the data and report fitted single-index coefficients for each variable.

```
library(SIQR)
#load data from MASS
library(MASS)
medv<- Boston$medv
RM <- Boston$rm
logTAX <- log(Boston$tax)
PTRATIO <- Boston$ptratio
logLSTAT <- log(Boston$lstat)

X <- cbind(RM,logTAX,PTRATIO,logLSTAT)
y0 <- medv - mean(medv)
beta0 <- NULL
tau.vec <- c(0.25,0.50,0.75)
est.coefficient <- matrix(NA, nrow = length(tau.vec), ncol = 5)
est.coefficient[,1] <- tau.vec
for (i in 1:length(tau.vec)){
est <- siqr(y0,X,beta.initial = beta0, tau=tau.vec[i],maxiter = 30,tol = 1e-8)
est.coefficient[i,2:5] <- est$beta
}
colnames(est.coefficient) <- c("quantile tau",colnames(X))
est.coefficient
```

```
#>      quantile tau        RM     logTAX     PTRATIO   logLSTAT
#> [1,]         0.25 0.3358285 -0.5243025 -0.06856117 -0.7795033
#> [2,]         0.50 0.3129182 -0.4294159 -0.06640472 -0.8445558
#> [3,]         0.75 0.2385613 -0.1933015 -0.07860687 -0.9484429
```

The estimated 0.25, 0.50, and 0.75 quantiles and their 95% pointwise confidence bounds are plotted with the following codes and outputs.

```
est.tau25 <- siqr(y0,X,beta.initial = NULL, tau=0.25)
plot.siqr(est.tau25,bootstrap.interval = TRUE)
```

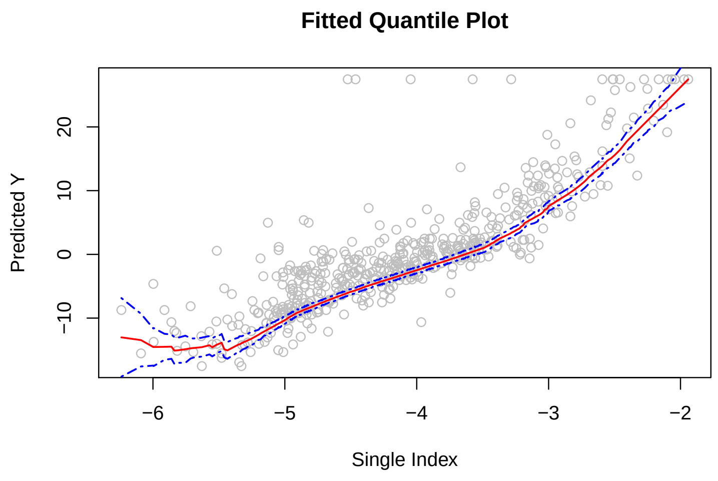


Figure 1: The R output of plot.siqr with estimated 0.25 quantiles and the 95% pointwise confidence bounds.

```
est.tau50 <- siqr(y0,X,beta.initial = NULL, tau=0.50)
plot.siqr(est.tau05,bootstrap.interval = TRUE)
```

```
est.tau75 <- siqr(y0,X,beta.initial = NULL, tau=0.75)
plot.siqr(est.tau75,bootstrap.interval = TRUE)
```

As the estimated single-index function curves are almost monotonically increasing across different quantiles, variables that contribute positively to the single index affect the response variable (medv) positively. Based on the estimated coefficients and above plots, we found that the number of rooms

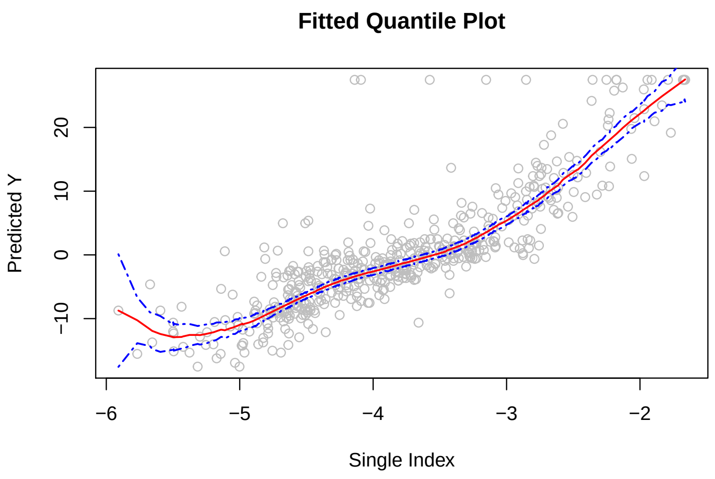


Figure 2: The R output of `plot.siqr` with estimated 0.50 quantiles and the 95% pointwise confidence bounds.

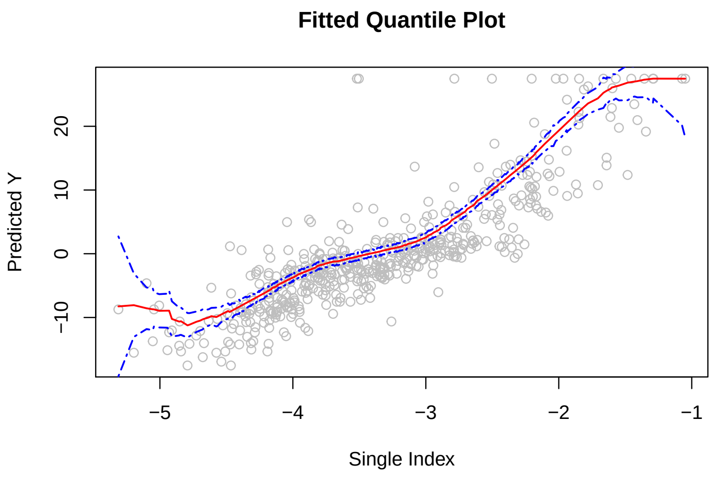


Figure 3: The R output of `plot.siqr` with estimated 0.75 quantiles and the 95% pointwise confidence bounds.

per house (rm) positively affects different quantiles. This matches the intuition that people value large spaces and multi-functional rooms. The property tax rate ln(tax) has a negative impact on housing prices across different quantiles. However, the influence of the tax rate is not significant at higher quantile $\tau = 0.75$. That suggests the tax rate may be less concerned for higher-income households, possibly due to tax deduction towards their income tax. Both the pupil-teacher ratio (ptratio) and the percentage of the lower (educational) status of the population ln(lstat) show negative influences on housing values, especially for the higher quantiles. It may suggest that potential buyers prefer areas featuring solid educational resources for their children and neighbors with higher education degrees and that preference grows more vital for more expensive houses.

Simulation

We consider two simulation settings. In the first simulation example, we use a sine-bump model with homoscedastic errors:

$$y = 5\sin\left(\frac{\pi\left(\mathbf{x}\boldsymbol{\beta} - A\right)}{C - A}\right) + 0.1Z, \tag{10}$$

where $A = \frac{\sqrt{3}}{2} - \frac{1.645}{\sqrt{12}}, C = \frac{\sqrt{3}}{2} + \frac{1.645}{\sqrt{12}}$, $\mathbf{x}$ is an $n \times 3$ design matrix that draws from an independent uniform distribution with min of 0 and max of 1, and the residual Z follows a standard normal distribution. The true single-index parameter $\boldsymbol{\beta} = (1,1,1)^\intercal/\sqrt{3}$.

	Estimate	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$
	mean	0.5782	0.5727	0.5725
$\tau = 0.25$	s.e.	0.0131	0.0281	0.0293
	bias	0.0009	-0.0046	-0.0048
	mean	0.5787	0.5755	0.5774
$\tau = 0.50$	s.e.	0.0115	0.0105	0.0111
	bias	0.0014	-0.0018	0.0003
	mean	0.5803	0.5756	0.5757
$\tau = 0.75$	s.e.	0.0119	0.0110	0.0118
	bias	0.0029	-0.0017	-0.0016

Table 1: Summary of parameter estimates for sine-bump simulation example 1 of sample size $n = 400$. True $\boldsymbol{\beta} = (1,1,1)^\intercal/\sqrt{3}$. The sample mean, standard error (s.e.), and bias of the parameter estimates of single-index coefficients from 200 replications.

The single-index coefficients are estimated via a series of quantile regressions with $\tau = 0.25, 0.50, 0.75$. Table 1 reports the mean, standard error (s.e.), and bias for each parameter estimate with sample size $n = 400$ over $M = 200$ replications on the simulation example 1. One can see that the algorithm for our R package **SIQR** is effective as the estimates are close to the true values.

For demonstration purposes, we show codes to generate data from (10) and fit the SIQR model using $\tau = 0.50$ with 200 replications as follows:

```
n <- 400
beta0 <- c(1, 1, 1)/sqrt(3)
n.sim <- 200
tau <- 0.50
data <- generate.data(n, true.theta=beta0, setting = "setting1",ncopy = n.sim)
sim.results.50 <- foreach(m = 1:n.sim,.combine = "rbind") %do% {
X <- data$X
Y <- data$Y[[m]]
est <- siqr(Y, X, beta.initial = c(2,1,0), tau=0.50,maxiter = 30,tol = 1e-8)
return(est$beta)
}
```

Note that this process has been repeated for the cases with $\tau = 0.25, 0.75$. We obtain a box plot of estimated single-index coefficients for $\tau = 0.25, 0.50, 0.75$, respectively, by applying the following code snippet.

```
boxplot(data.frame((sim.results.25)), outline=T,notch=T,range=1,
main = "Boxplots of Coefficient Estimates, tau = 0.25",horizontal = F)


boxplot(data.frame((sim.results.50)), outline=T,notch=T,range=1,
main = "Boxplots of Coefficient , tau = 0.50",horizontal = F)


boxplot(data.frame((sim.results.75)), outline=T,notch=T,range=1,
main = "Boxplots of Coefficient Estimates, tau = 0.75",horizontal = F)
```

Next, we consider a location-scale model as simulation example 2, where both the location and the scale depend on a common index $u = \mathbf{x}\boldsymbol{\beta}$. The quantiles are "almost-linear-in-index" as in Yu and Jones (1998) when the single index u is close to zero:

$$y = 5\cos(\mathbf{x}\boldsymbol{\beta}) + \exp(-(\mathbf{x}\boldsymbol{\beta})^2) + E, \tag{11}$$

where $\mathbf{x}$ is an $n \times 2$ design matrix that draws from an independent normal distribution with a standard deviation of 0.25, and the residual E follows an exponential distribution with a mean 2. The single-index parameter $\boldsymbol{\beta} = (1,2)^\intercal/\sqrt{5}$.

The simulated data are generated with the following codes. The sample size $n = 400$ with 100 replications. We only present the case when $\tau = 0.50$ for demonstration purposes.

```
n <- 400
beta0 <- c(1, 2)/sqrt(5)
```

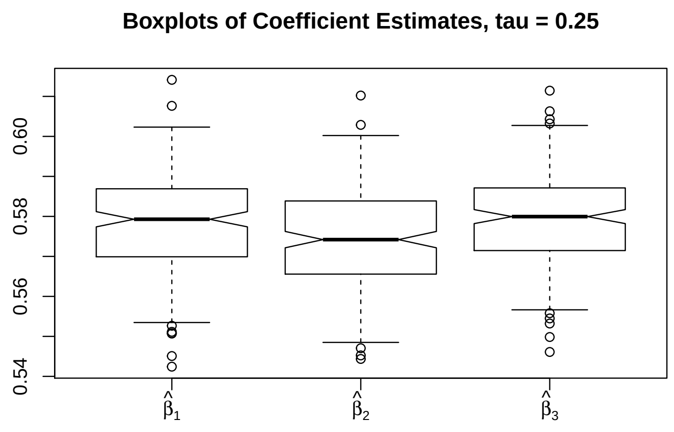


Figure 4: The box plot of estimated single-index coefficients for $\tau = 0.25$ from example 1.

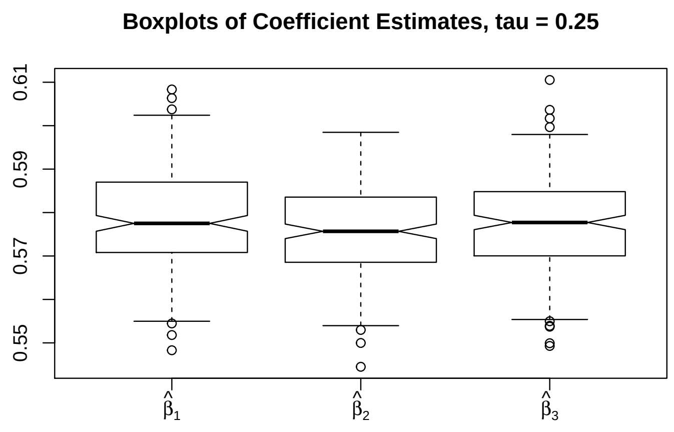


Figure 5: The box plot of estimated single-index coefficients for $\tau = 0.50$ from example 1.

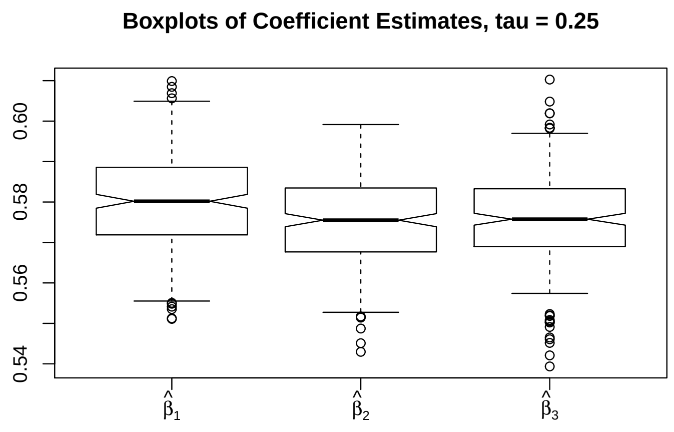


Figure 6: The box plot of estimated single-index coefficients for $\tau = 0.75$ from example 1.

```
n.sim <- 100
tau <- 0.5
data <- generate.data(n, true.theta=beta0, setting = "setting3",ncopy = n.sim)
sim.results <- foreach(m = 1:n.sim,.combine = "rbind") %do% {
X <- data$X
Y <- data$Y[[m]]
est <- siqr(Y, X, beta.initial = NULL, tau=tau,maxiter = 30,tol = 1e-8)
est$beta
}
est.mean <- c(tau,apply(sim.results,2,mean))
names(est.mean) <- c("tau","beta1.hat","beta2.hat")
est.mean

est.mean <- cbind(p_vec,apply(sim_results,c(1,2),sd))
colnames(est.mean) <- c("quantile tau","X1","X2","X3")
est.mean

#> tau beta1.hat beta2.hat
#> 0.5 0.4515909 0.8917233
```

The average estimated single-index coefficients shown above are close to the true single-index parameter $\boldsymbol{\beta} = (1,2)^\intercal/\sqrt{5} \approx (0.4472, 0.8944)$. On top of that, the simulation standard error is also reported as below:

```
est.se <- c(tau,apply(sim.results,2,sd))
names(est.se) <- c("tau","beta1.se.hat","beta1.se.hat")
est.se

#>   tau beta1.se.hat beta1.se.hat
#>   0.5   0.02682211   0.01359602
```

Meanwhile, the following box plots show that the estimated single-index coefficients are close to the true parameters with small deviations.

```
boxplot(data.frame((sim.results)), outline=T,notch=T,range=1,
main = "Boxplots of Coefficient Estimates (100 replications)",horizontal = F)
```

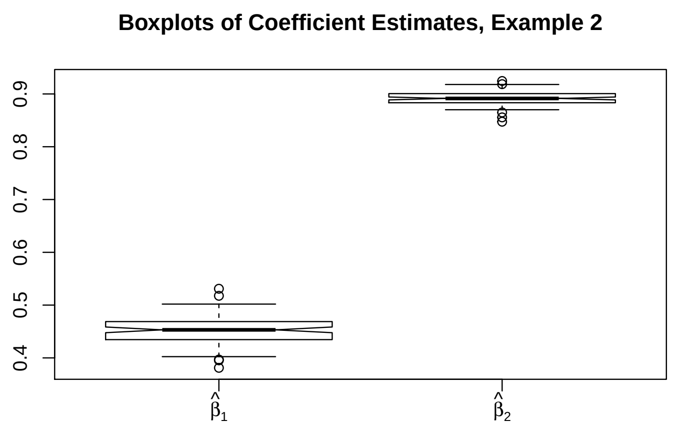


Figure 7: The box plot of estimated single-index coefficients for $\tau = 0.50$ from example 2.

Similarly, we plot the estimated quantiles and their 95% pointwise confidence bounds with the provided plot function `plot.siqr`. The observed data points are also plotted.

```
est.sim.50 <- siqr(data$Y[[1]],data$X,beta.initial = NULL, tau=0.5)
plot.siqr(est.sim.50,bootstrap.interval = TRUE)
```

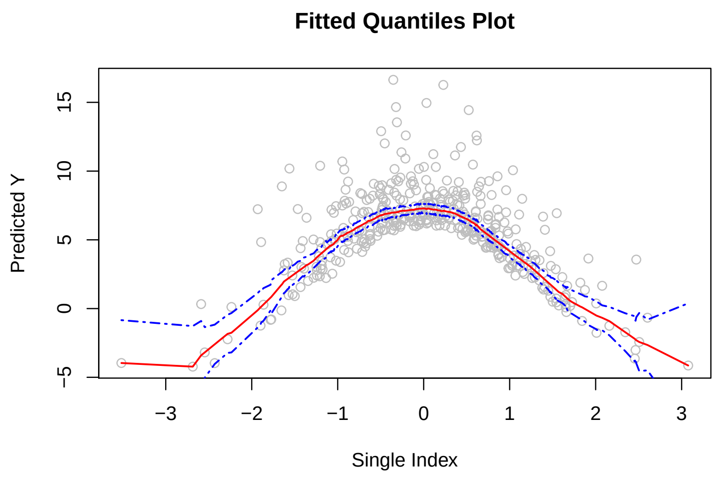


Figure 8: The R output of `plot.siqr` with estimated 0.50 quantiles and the 95% pointwise confidence bounds from example 2.

Summary

In this paper, we present the R package **SIQR** for the local linear approach to single-index quantile regression models in Wu et al. (2010). We demonstrate the package applications to a popular Boston-housing data application and two simulation studies. It is our hope that the package will be useful to a variety of applications, especially for complex heterogeneous data where flexible quantile regression modeling is desirable.

Bibliography

P. Chaudhuri, K. Doksum, and A. Samarov. On average derivative quantile regression. *Annals of Statistics*, 25(2):715–744, Apr. 1997. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1031833670. URL `https://projecteuclid.org/euclid.aos/1031833670`. Publisher: Institute of Mathematical Statistics. [p460]

R. Koenker and G. Bassett. Regression Quantiles. *Econometrica*, 46(1):33–50, 1978. ISSN 0012-9682. doi: 10.2307/1913643. URL `https://www.jstor.org/stable/1913643`. Publisher: [Wiley, Econometric Society]. [p460, 461]

E. Kong and Y. Xia. A SINGLE-INDEX QUANTILE REGRESSION MODEL AND ITS ESTIMATION. *Econometric Theory*, 28(4):730–768, 2012. ISSN 0266-4666. URL `http://www.jstor.org/stable/23257656`. Publisher: Cambridge University Press. [p460]

J. D. Opsomer and D. Ruppert. A Fully Automated Bandwidth Selection Method for Fitting Additive Models. *Journal of the American Statistical Association*, 93(442):605–619, 1998. ISSN 0162-1459. doi: 10.2307/2670112. URL `http://www.jstor.org/stable/2670112`. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p463]

D. Ruppert, S. J. Sheather, and M. P. Wand. An Effective Bandwidth Selector for Local Least Squares Regression. *Journal of the American Statistical Association*, 90(432):1257–1270, 1995. ISSN 0162-1459. doi: 10.2307/2291516. URL `http://www.jstor.org/stable/2291516`. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p461]

T. Z. Wu, K. Yu, and Y. Yu. Single-index quantile regression. *Journal of Multivariate Analysis*, 101(7):1607–1621, Aug. 2010. ISSN 0047-259X. doi: 10.1016/j.jmva.2010.02.003. URL `http://www.sciencedirect.com/science/article/pii/S0047259X10000333`. [p460, 461, 462, 463, 469]

Y. Xia and W. Härdle. Semi-parametric estimation of partially linear single-index models. *Journal of Multivariate Analysis*, 97(5):1162–1184, May 2006. ISSN 0047-259X. doi: 10.1016/j.jmva.2005.11.005. URL `http://www.sciencedirect.com/science/article/pii/S0047259X05001995`. [p462]

K. Yu and M. C. Jones. Local Linear Quantile Regression. *Journal of the American Statistical Association*, 93(441):228–237, 1998. ISSN 0162-1459. doi: 10.2307/2669619. URL http://www.jstor.org/stable/2669619. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p461, 466]

K. Yu and Z. Lu. Local Linear Additive Quantile Regression. *Scandinavian Journal of Statistics*, 31(3): 333–346, 2004. ISSN 0303-6898. URL http://www.jstor.org/stable/4616834. Publisher: [Board of the Foundation of the Scandinavian Journal of Statistics, Wiley]. [p460, 463]

K. Yu, Z. Lu, and J. Stander. Quantile Regression: Applications and Current Research Areas. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 52(3):331–350, 2003. ISSN 0039-0526. URL http://www.jstor.org/stable/4128208. Publisher: [Royal Statistical Society, Wiley]. [p460]

Y. Yu and D. Ruppert. Penalized Spline Estimation for Partially Linear Single-Index Models. *Journal of the American Statistical Association*, 97(460):1042–1054, 2002. ISSN 0162-1459. URL https://www.jstor.org/stable/3085829. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p461]

Tianhai Zu
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
(ORCiD if desired)
zuti@mail.uc.edu

Yan Yu
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
https://orcid.org/0000-0002-2859-3093
Yan.YU@uc.edu

mgee2: An R package for marginal analysis of longitudinal ordinal data with misclassified responses and covariates

by Yuliang Xu, Shuo Shuo Liu and Grace Y. Yi

Abstract Marginal methods have been widely used for analyzing longitudinal ordinal data due to their simplicity in model assumptions, robustness in inference results, and easiness in the implementation. However, they are often inapplicable in the presence of measurement errors in the variables. Under the setup of longitudinal studies with ordinal responses and covariates subject to misclassification, Chen et al. (2014) developed marginal methods for misclassification adjustments using the second-order estimating equations and proposed a two-stage estimation approach when the validation subsample is available. Parameter estimation is conducted through the Newton-Raphson algorithm, and the asymptotic distribution of the estimators is established. While the methods of Chen et al. (2014) can successfully correct the misclassification effects, its implementation is not accessible to general users due to the lack of a software package. In this paper, we develop an R package, *mgee2*, to implement the marginal methods proposed by Chen et al. (2014). To evaluate the performance and illustrate the features of the package, we conduct numerical studies.

Introduction

Analysis of longitudinal ordinal data is a common research topic in health science and survey sampling. Typically, Liang and Zeger (1986) introduced the generalized estimating equations (GEE) method that gave consistent estimation with mild assumptions of the joint distribution of the repeated measurements. This method has been used widely in analyzing longitudinal binary and categorical data. The validity of the GEE method hinges on the critical condition that data are precisely observed, which is commonly infeasible and violated in practice (Yi, 2017). Extensive discussions about covariate error (Carroll et al., 2006) and response with binary misclassification (Neuhaus, 1999; Chen et al., 2011; Yi, 2017) have been conducted in the literature. For example, Neuhaus (1999) investigated the bias due to errors in the response. Yi (2008) proposed a simulation–extrapolation (SIMEX) method to handle both dropout and covariate measurement error problems in longitudinal studies. Furthermore, in Yi (2017, Ch5), the impact of covariate measurement error on longitudinal data analysis was investigated, and methods of addressing covariate measurement error effects were described.

To accommodate effects induced from error-prone correlated ordinal responses and ordinal covariates simultaneously, Chen et al. (2014) proposed GEE-based methods for the estimation of both mean and association parameters. The proposed methods are based on formulating unbiased second-order estimating functions and solving the resulting equations using the Newton-Raphson algorithm. The asymptotic distributions for the proposed estimators are established. While the methods of Chen et al. (2014) correct for error effects due to misclassified variables, the methods cannot be used by the analysts without programming the implementation procedures. To expedite the use of the methods for problems in applications, in this paper, we develop an R package, called *mgee2*, to implement the methods of Chen et al. (2014).

Our work offers an R package complement to available R packages for analyzing longitudinal data with misclassified observations. It is relevant to but differs from available R packages about measurement error. For example, the package *SAMBA*, developed by Beesley and Mukherjee (2020), provides resources for fitting logistic regression with misclassified binary outcomes. The R package *misclassGLM* implements inferential procedures for generalized linear models with misclassified covariates proposed by Dlugosz et al. (2017); Zhang and Yi (2019) developed the package *augSIMEX* to implement the method proposed by Yi et al. (2015) for fitting generalized linear models with mixed continuous and discrete covariates subject to mismeasurement.

When the degree of measurement error is very severe, the observed surrogate measurements are virtually useless, and hence the corresponding variables may be alternatively treated as subject to missingness. Regarding the analysis of longitudinal data with missing observations, packages *kml* and *kml3d*, developed by Genolini et al. (2015), describe the implementation procedures of k-means for longitudinal clustered data with missing observations. Carey (2015) developed the package *gee* to solve generalized estimation equations with longitudinal data missing completely at random. Xu et al. (2018) developed the package *wgeesel* for using weighted generalized estimating equations approaches to analyze longitudinal clustered data with data missing at random. Xiong and Yi (2019) developed the package *swgee* for analyzing longitudinal data with missingness in the response and measurement error in covariates. Our package *mgee2* differs from those packages in its ability to simultaneously handle the features of misclassification in correlated ordinal responses and ordinal covariates, which to our best knowledge, is the first software package to address this problem.

The article is organized as follows. Section *Model setup* introduces the notations and estimation procedures proposed by Chen et al. (2014). Section *Package details* describes the usage of the package *mgee2*. Section *Data analysis* illustrates the package by simulation studies and a real dataset. We finally conclude the article in Section *Summary*.

Model setup

We first review the notation and formulations of Chen et al. (2014). For $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$, let Y_{ij} denote an error-prone ordinal response variable for subject i at visit j. Suppose that the response variable Y_{ij} has $(K+1)$ categories, denoted $0, 1, \ldots, K$, and that an error-prone ordinal covariate $\boldsymbol{X}_{ij}$ has $(H+1)$ categories, denoted $0, 1, \ldots, H$. Let $\boldsymbol{X}_{ij} = \left(X_{ij1}, \ldots, X_{ijH}\right)^T$ be the misclassification-prone vector of binary variables such that $X_{ijq} = I$(the covariate $\boldsymbol{X}_{ij}$ in category q) for $q = 0, 1, \ldots, H$, and let $\boldsymbol{Z}_{ij}$ be the vector of covariates that are free of measurement error, where $I(\cdot)$ is the indicator function. Furthermore, we define $\boldsymbol{X}_i = \left(\boldsymbol{X}_{i1}^T, \ldots, \boldsymbol{X}_{im_i}^T\right)^T$ and $\boldsymbol{Z}_i = \left(\boldsymbol{Z}_{i1}^T, \ldots, \boldsymbol{Z}_{im_i}^T\right)^T$.

Response process

Let
$$\lambda_{ijk} = P\left(\mathsf{Y}_{ij} \geq k | \boldsymbol{X}_i, \boldsymbol{Z}_i\right) \tag{1}$$
be the univariate cumulative probability with $k = 1, \ldots, K$, and adopt the assumption $P\left(\mathsf{Y}_{ij} \geq k | \boldsymbol{X}_i, \boldsymbol{Z}_i\right) = P\left(\mathsf{Y}_{ij} \geq k | \boldsymbol{X}_{ij}, \boldsymbol{Z}_{ij}\right)$ (Pepe and Anderson, 1994). Consider the proportional odds models
$$\text{logit } \lambda_{ijk} = \boldsymbol{\beta}_{0k} + \boldsymbol{X}_{ij}^T \boldsymbol{\beta}_x + \boldsymbol{Z}_{ij}^T \boldsymbol{\beta}_z,$$
where $\boldsymbol{\beta}_{0k}$, $\boldsymbol{\beta}_x$, and $\boldsymbol{\beta}_z$ are regression parameters, $k = 1, \ldots, K$, $j = 1, \ldots, m_i$, and $i = 1, \ldots, n$. Similar to Williamson et al. (1995), we measure the association between a pair of responses for the same subject at two different visits by the global odds ratio
$$\psi_{i,jk,j'k'} = \frac{P\left(\mathsf{Y}_{ij} \geq k, \mathsf{Y}_{ij'} \geq k' | \boldsymbol{X}_i, \boldsymbol{Z}_i\right) \times P\left(\mathsf{Y}_{ij} < k, \mathsf{Y}_{ij'} < k' | \boldsymbol{X}_i, \boldsymbol{Z}_i\right)}{P\left(\mathsf{Y}_{ij} \geq k, \mathsf{Y}_{ij'} < k' | \boldsymbol{X}_i, \boldsymbol{Z}_i\right) \times P\left(\mathsf{Y}_{ij} < k, \mathsf{Y}_{ij} \geq k' | \boldsymbol{X}_i, \boldsymbol{Z}_i\right)}, \tag{2}$$
where $k, k' = 1, \ldots, K$, and $j \neq j'$. To characterize the dependence of the global odds ratios on covariates, the log-linear models can be expressed as
$$\log\ \psi_{i,jk,j'k'} = \phi + \phi_k + \phi_{k'} + \phi_{kk'} + \mathbf{u}_{ijj'}^T \boldsymbol{\alpha}_1,$$
where ϕ is the global intercept, ϕ_k and $\phi_{k'}$ correspond to the effect of category k and of category k', respectively, $\phi_{kk'}$ is the interaction effect between categories k and k' with $\phi_{kk'} = \phi_{k'k}$, and $\boldsymbol{\alpha}_1$ is a vector of parameters corresponding to pair-specific covariates, denoted $\mathbf{u}_{ijj'}$. The constraint $\phi_1 = \phi_{1k} = \phi_{k1} = 0$ is set for the model identification for $k = 1, \ldots, K$ (Williamson et al., 1995).

Let $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_{0k}^T, \boldsymbol{\beta}_x^T, \boldsymbol{\beta}_z^T\right)^T$, $\boldsymbol{\alpha} = \left(\phi, \phi_k, \phi_{kk'}, \boldsymbol{\alpha}_1^T\right)^T$, and $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^T, \boldsymbol{\alpha}^T\right)^T$. For $k = 1, \ldots, K$, let $\mathbf{Y}_{ij} = \left(\mathsf{Y}_{ij1}, \ldots, \mathsf{Y}_{ijk}\right)^T$ with $\mathsf{Y}_{ijk} = I\left(\mathsf{Y}_{ij} = k\right)$. Define $\mathbf{Y}_i = \left(\mathbf{Y}_{i1}^T, \ldots, \mathbf{Y}_{im_i}^T\right)^T$. For $j < j'$, let $C_{i,jk,j'k'} = \mathsf{Y}_{ijk}\mathsf{Y}_{ij'k'}$, $\boldsymbol{C}_{ijj'} = \left(C_{i,j1,j'1}, C_{i,j1,j'2}, \ldots, C_{i,jK,j'K'}\right)^T$, and $\boldsymbol{C}_i = \left(\boldsymbol{C}_{ijj'}^T, j < j'\right)^T$. The univariate and bivariate marginals, $\boldsymbol{\mu}_i = E\left(\mathbf{Y}_i | \boldsymbol{X}_i, \boldsymbol{Z}_i\right)$ and $\boldsymbol{\xi}_i = E\left(\boldsymbol{C}_i | \boldsymbol{X}_i, \boldsymbol{Z}_i\right)$, can be expressed in terms of the global odds ratios and univariate and bivariate cumulative probabilities; the detailed expressions are given by Chen et al. (2014).

As a result, the estimating functions for the mean and association parameters $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are given by
$$\boldsymbol{U}_{1i}\left(\boldsymbol{\theta}; \mathbf{Y}_i, \boldsymbol{X}_i, \boldsymbol{Z}_i\right) = \boldsymbol{D}_{1i} \boldsymbol{V}_{1i}^{-1} \left(\mathbf{Y}_i - \boldsymbol{\mu}_i\right) \tag{3}$$
and
$$\boldsymbol{U}_{2i}\left(\boldsymbol{\theta}; \mathbf{Y}_i, \boldsymbol{X}_i, \boldsymbol{Z}_i\right) = \boldsymbol{D}_{2i} \boldsymbol{V}_{2i}^{-1} \left(\boldsymbol{C}_i - \boldsymbol{\xi}_i\right), \tag{4}$$
respectively, where $\boldsymbol{D}_{1i} = \partial \boldsymbol{\mu}_i^T / \partial \boldsymbol{\beta}$, $\boldsymbol{D}_{2i} = \partial \boldsymbol{\xi}_i^T / \partial \boldsymbol{\alpha}$, and $\boldsymbol{V}_{1i}$ and $\boldsymbol{V}_{2i}$ are the conditional covariance

matrices for $\mathbf{Y}_i$ and $\boldsymbol{C}_i$, given $\boldsymbol{X}_i$ and $\boldsymbol{Z}_i$.

Case 1: Estimation with known misclassification probabilities

If the true measurements of the responses and covariates are available, (3) and (4) can be used for estimation of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$. However, in applications, the $\mathbf{Y}_{ij}$ and the $\boldsymbol{X}_{ij}$ may be subject to misclassification. Let $\boldsymbol{S}_{ij}$ and $\boldsymbol{W}_{ij}$ be surrogate measurements of $\mathbf{Y}_{ij}$ and $\boldsymbol{X}_{ij}$, respectively. Let $\tau_{ijkl} = P\left(\boldsymbol{S}_{ij} = l|\mathbf{Y}_{ij} = k, \boldsymbol{Z}_i\right)$ be the conditional probability concerning the response for subject i at visit j where $k, l = 0, \ldots, K$. Let $\pi_{ijqr} = P\left(\boldsymbol{W}_{ij} = r|\boldsymbol{X}_{ij} = q, \boldsymbol{Z}_i\right)$ be the conditional probability concerning the covariate for subject i at visit j where $q, r = 0, \ldots, H$. Consider the generalized logistic models by setting category 0 as the reference:

$$\log\left(\tau_{ijkl}/\tau_{ijk0}\right) = \boldsymbol{L}_{ij}^T \gamma_{kl} \quad \text{for} \quad l = 1, \ldots, K; k = 0, \ldots, K$$

and

$$\log\left(\pi_{ijqr}/\pi_{ijq0}\right) = \boldsymbol{L}_{ij}^{xT} \boldsymbol{\varphi}_{qr} \quad \text{for} \quad r = 1, \ldots, H; q = 0, \ldots, H,$$

where $\boldsymbol{L}_{ij}$ and $\boldsymbol{L}_{ij}^x$ are vectors of variables related to response and covariate misclassification processes, respectively, and γ_{kl} and $\boldsymbol{\varphi}_{qr}$ are vectors of regression parameters.

Let $\gamma_k = (\gamma_{k1}^T, \ldots, \gamma_{kK}^T)^T$ and $\gamma = (\gamma_0^T, \ldots, \gamma_K^T)^T$. Let $\boldsymbol{\varphi}_q = \left(\boldsymbol{\varphi}_{q1}^T, \ldots, \boldsymbol{\varphi}_{qH}^T\right)^T$ and $\boldsymbol{\varphi} = (\boldsymbol{\varphi}_0^T, \ldots, \boldsymbol{\varphi}_H^T)^T$. Let $\boldsymbol{\eta} = \left(\gamma^T, \boldsymbol{\varphi}^T\right)^T$. Define the $K \times K$ matrix $\boldsymbol{R}_{ij} = \left(\boldsymbol{\tau}_{ij1} - \boldsymbol{\tau}_{ij0}, \ldots, \boldsymbol{\tau}_{ijK} - \boldsymbol{\tau}_{ij0}\right)$ and the $H \times H$ matrix $\boldsymbol{G}_{ij} = \left(\boldsymbol{\pi}_{ij1} - \boldsymbol{\pi}_{ij0}, \ldots, \boldsymbol{\pi}_{ijK} - \boldsymbol{\pi}_{ij0}\right)$, where $\boldsymbol{\tau}_{ijk} = \left(\tau_{ijk1}, \ldots, \tau_{ijkK}\right)^T$ and $\boldsymbol{\pi}_{ijk} = \left(\pi_{ijk1}, \ldots, \pi_{ijkK}\right)^T$. Then the unbiased surrogates for $\mathbf{Y}_{ij}$ and $\boldsymbol{X}_{ij}$ are constructed, respectively, by

$$\mathbf{Y}_{ij}^* = \boldsymbol{R}_{ij}^{-1}\left(\boldsymbol{S}_{ij} - \boldsymbol{\tau}_{ij0}\right)$$

and

$$\boldsymbol{X}_{ij}^* = \boldsymbol{G}_{ij}^{-1}\left(\boldsymbol{W}_{ij} - \boldsymbol{\pi}_{ij0}\right),$$

where we write $\mathbf{Y}_{ij}^* = \left(Y_{ij1}^*, \ldots, Y_{ijK}^*\right)^T$, $\boldsymbol{X}_{ij}^* = \left(X_{ij1}^*, \ldots, X_{ijK}^*\right)^T$, and let $\mathbf{Y}_i^* = \left(\mathbf{Y}_{i1}^T, \ldots, \mathbf{Y}_{im_i}^{*T}\right)^T$. Let $\boldsymbol{e}_q$ be an H-dimensional vector whose rth element is an indicator $I(r = q)$ for $q = 1, \ldots, H$ and let $\boldsymbol{e}_0 = 0$.

If $\boldsymbol{\eta}$ is known, then

$$\boldsymbol{U}_{1i}^*(\boldsymbol{\theta}) = \sum_{q_{m_i}=0}^{H} \cdots \sum_{q_1=0}^{H} \left[\boldsymbol{U}_{1i}\left\{\theta; \mathbf{Y}_i^*, \left(\boldsymbol{e}_{q_1}^T, \ldots, \boldsymbol{e}_{q_{i_i}}^T\right)^T, \boldsymbol{Z}_i\right\} \prod_{j=1}^{m_i} X_{ijq_j}^*\right]$$

and

$$\boldsymbol{U}_{2i}^*(\boldsymbol{\theta}) = \sum_{q_{m_i}=0}^{H} \cdots \sum_{q_1=0}^{H} \left[\boldsymbol{U}_{2i}\left\{\theta; \mathbf{Y}_i^*, \left(\boldsymbol{e}_{q_1}^T, \ldots, \boldsymbol{e}_{q_{i_i}}^T\right)^T, \boldsymbol{Z}_i\right\} \prod_{j=1}^{m_i} X_{ijq_j}^*\right]$$

are unbiased estimating functions of $\boldsymbol{\theta}$, as shown in Appendix 2 of Chen et al. (2014). Estimation of $\boldsymbol{\theta}$ can then be obtained by solving

$$\sum_{i=1}^{n} \left\{ \begin{array}{c} \boldsymbol{U}_{1\mathrm{i}}^*(\boldsymbol{\theta}) \\ \boldsymbol{U}_{2i}^*(\boldsymbol{\theta}) \end{array} \right\} = \mathbf{0} \tag{5}$$

for $\boldsymbol{\theta}$.

Case 2: Estimation with validation data

Case 1 highlights the estimation of $\boldsymbol{\theta}$ when the parameter $\boldsymbol{\eta}$ for the misclassification models is known or specified as a given value. In applications, $\boldsymbol{\eta}$ is unknown and may be estimated from a validation subsample. In this case, we modify the estimation procedure based on (5) and describe a two-stage estimation procedure. First, let $\delta_{ij} = I$(subject i at visit j is included in the validation subsample). Using validation data (i.e., $\delta_{ij} = 1$), we may estimate τ_{ij} and π_{ij}.

Define $\boldsymbol{D}_{\gamma ij} = \partial\boldsymbol{\tau}_{ij}^T/\partial\gamma$ and $\boldsymbol{D}_{\boldsymbol{\varphi}ij} = \partial\boldsymbol{\pi}_{ij}^T/\partial\boldsymbol{\varphi}$, then estimating functions for γ and $\boldsymbol{\varphi}$ are given by $\boldsymbol{Q}_{\gamma i}(\gamma) = \sum_{j=1}^{m_i} \boldsymbol{D}_{\gamma ij}\boldsymbol{V}_{\gamma ij}^{-1}\left\{\boldsymbol{S}_{ij} - \boldsymbol{\tau}_{ij}\right\}\delta_{ij}$ and $\boldsymbol{Q}_{\varphi i}(\boldsymbol{\varphi}) = \sum_{j=1}^{m_i} \boldsymbol{D}_{\varphi ij}\boldsymbol{V}_{\varphi ij}^{-1}\left\{\boldsymbol{W}_{ij} - \boldsymbol{\pi}_{ij}\right\}\delta_{ij}$, where $\boldsymbol{V}_{\gamma ij}$ and $\boldsymbol{V}_{\boldsymbol{\varphi}ij}$ are, respectively, the conditional covariance matrix for $\boldsymbol{S}_{ij}$ and $\boldsymbol{W}_{ij}$, given $\mathbf{Y}_{ij}$ and the true covariates.

Let $\tilde{Y}_{ijk} = Y_{ijk}$ if $\delta_{ij} = 1$ and $\tilde{Y}_{ijk} = Y^*_{ijk}$ otherwise, then we write $\tilde{\mathbf{Y}}_{ij} = \left(\tilde{Y}_{ij1}, \ldots, \tilde{Y}_{ijK}\right)^T$. Let $\tilde{X}_{ijq} = X_{ijq}$ if $\delta_{ij} = 1$ and $\tilde{X}_{ijq} = X^*_{ijq}$ otherwise. Then the augmented estimating functions of $\boldsymbol{\theta}$ are given by

$$\tilde{\boldsymbol{U}}_{1i}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \sum_{q_{m_i}=0}^{H} \cdots \sum_{q_1=0}^{H} \left[\boldsymbol{U}_{1i} \left\{ \theta; \tilde{\mathbf{Y}}_i, \left(\boldsymbol{e}^T_{q_1}, \ldots, \boldsymbol{e}^T_{q_m}\right)^T, \boldsymbol{Z}_i \right\} \prod_{j=1}^{m_i} \tilde{X}_{ijq_j} \right] \tag{6}$$

and

$$\tilde{\boldsymbol{U}}_{2i}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \sum_{q_{m_i}=0}^{H} \cdots \sum_{q_1=0}^{H} \left[\boldsymbol{U}_{2i} \left\{ \theta; \tilde{\mathbf{Y}}_i, \left(\boldsymbol{e}^T_{q_1}, \ldots, \boldsymbol{e}^T_{q_m}\right)^T, \boldsymbol{Z}_i \right\} \prod_{j=1}^{m_i} \tilde{X}_{ijq_j} \right]. \tag{7}$$

Consequently, estimation of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ can be carried out by the two-stage procedure.

Stage 1. Solve $\sum_{i=1}^n \left\{ \begin{array}{c} \boldsymbol{Q}_{\gamma i}(\gamma) \\ \boldsymbol{Q}_{\varphi i}(\boldsymbol{\varphi}) \end{array} \right\} = \mathbf{0}$ for γ and $\boldsymbol{\varphi}$ and write $\hat{\boldsymbol{\eta}} = \left(\hat{\gamma}^T, \hat{\boldsymbol{\varphi}}^T\right)^T$, where $\hat{\gamma}$ and $\hat{\boldsymbol{\varphi}}$ are the estimators for γ and $\boldsymbol{\varphi}$, respectively.

Stage 2. Substitute $\boldsymbol{\eta}$ with $\hat{\boldsymbol{\eta}}$ in (6) and (7) and solve $\sum_{i=1}^n \left\{ \begin{array}{c} \tilde{\boldsymbol{U}}_{1i}(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}) \\ \tilde{\boldsymbol{U}}_{2i}(\boldsymbol{\theta}, \hat{\boldsymbol{\eta}}) \end{array} \right\} = \mathbf{0}$ for $\boldsymbol{\theta}$. Let $\hat{\boldsymbol{\theta}} = \left(\hat{\boldsymbol{\beta}}^T, \hat{\boldsymbol{\alpha}}^T\right)^T$ denote the resulting estimator $\boldsymbol{\theta}$.

Chen et al. (2014) established the asymptotic distribution of $\hat{\boldsymbol{\theta}}$. Let $\tilde{\boldsymbol{U}}_i(\boldsymbol{\theta}, \boldsymbol{\eta}) = \{\tilde{\boldsymbol{U}}^T_{1i}(\boldsymbol{\theta}, \boldsymbol{\eta}), \tilde{\boldsymbol{U}}^T_{2i}(\boldsymbol{\theta}, \boldsymbol{\eta})\}^T$, $\boldsymbol{Q}_i(\boldsymbol{\eta}) = \left\{\boldsymbol{Q}^T_{\gamma i}(\gamma), \boldsymbol{Q}^T_{\varphi i}(\boldsymbol{\varphi})\right\}^T$, $\boldsymbol{\Omega}_i(\boldsymbol{\theta}, \boldsymbol{\eta}) = \tilde{\boldsymbol{U}}_i(\boldsymbol{\theta}, \boldsymbol{\eta}) - E\left\{\partial \tilde{U}_i(\boldsymbol{\theta}, \boldsymbol{\eta})/\partial \boldsymbol{\eta}^T\right\} \cdot \left[E\left\{\partial \boldsymbol{Q}_i(\boldsymbol{\eta})/\partial \boldsymbol{\eta}^T\right\}\right]^{-1}$, and $\tilde{\boldsymbol{\Gamma}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = E\left\{\partial \tilde{\boldsymbol{U}}_i(\boldsymbol{\theta}, \boldsymbol{\eta})/\partial \boldsymbol{\theta}^T\right\}$. Then, under regularity conditions, $n^{1/2}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})$ is asymptotically normally distributed with mean $\mathbf{0}$ and covariance matrix $\tilde{\boldsymbol{\Gamma}}^{-1}\tilde{\boldsymbol{\Sigma}}\left(\tilde{\boldsymbol{\Gamma}}^{-1}\right)^T$, where $\tilde{\boldsymbol{\Sigma}} = E\left\{\boldsymbol{\Omega}_i(\boldsymbol{\theta}, \boldsymbol{\eta})\boldsymbol{\Omega}^T_i(\boldsymbol{\theta}, \boldsymbol{\eta})\right\}$.

Package details

We develop an R package, called *mgee2*, to implement the misclassification adjustment method described in the preceding section. This package requires support from the external packages *MASS* (Venables and Ripley, 2002), *Matrix* (Bates and Maechler, 2019), and *ggplot2* (Wickham, 2016). Our *mgee2* package mainly contains two functions, `mgee2k` and `mgee2v`, respectively, implementing cases 1 and 2 described in the previous section. Specifically, `mgee2k` implements the method where the misclassification parameters are given, and `mgee2v` implements the misclassification method for the case where validation data are available to estimate misclassification probabilities. We now describe the details of these two functions.

`mgee2k`

`mgee2k` implements the misclassification adjustment method outlined in Case 1 of the previous section, where the misclassification parameters are known. In this case, validation data are not required, and only the observed data of the outcome and covariates are needed for the implementation.

The function `mgee2k` requires the data set to be grouped by the individual id, $i = 1, ..., n$, and each individual has m_i rows of data each corresponding to the visit time $j = 1, ..., m_i$. The column name of the individual id is indicated by the argument `id`. The misclassification matrices for the response and covariate variables are recorded by the arguments `gamMat` and `varphiMat`, respectively, which need to be specified by the user.

To call `mgee2k`, we issue the following command,

```
mgee2k(formula, id, data, corstr="exchangeable", misvariable,
             gamMat, varphiMat, maxit=50, tol=1e-3)
```

where the meaning of each argument is described as follows:

- `formula`: a formula object which specifies the relationship between the response and covariates for the observed data.
- `id`: a character object which records individual id in the data.
- `data`: a dataframe or matrix object for the observed data set.
- `corstr`: a character object. The default value is "exchangeable", corresponding to the structure where the association between two paired responses is considered to be a constant. The other option is "log-linear" which indicates the log-linear association between two paired responses.

- `misvariable`: a character object which names the error-prone covariate W.
- `maxit`: an integer which specifies the maximum number of iterations. The default is 50.
- `tol`: a numeric object which indicates the tolerance threshold. The default is 1e-3.
- `gamMat`: a matrix object which records the misclassification parameter γ for response `Y`.
- `varphiMat`: a matrix object which records the misclassification parameter ϕ for covariate X.

The function `mgee2k` returns a list of components:

- `beta`: the coefficients in the same order as that specified in the formula for the response and covariates.
- `alpha`: the coefficients for paired responses global odds ratios. The number of α coefficients corresponds to the paired responses odds ratio structure selected in `corstr`. When `corstr="exchangeable"`, only one baseline α is fitted.
- `variance`: the variance-covariance matrix of the estimators of all parameters.
- `convergence`: a logical variable; TRUE if the model converges.
- `iteration`: the number of iterations for the estimates of the model parameters to converge.
- `call`: an unevaluated function call which consists of the named function applied to the given arguments.

mgee2v

The function `mgee2v` does not require the misclassification parameters to be known, but requires the availability of validation data.

Similar to `mgee2k`, the function `mgee2v` needs the data set to be structured by individual id, $i = 1, ..., n$, and visit time, $j = 1, ..., m_i$. The data set should contain the observed response and covariates, S and W. To indicate whether or not a subject is in the validation set, an indicator variable `delta` should be added in the data set, and we use a column named `valid.sample.ind` for this purpose. The column name of the error-prone covariate W should also be specified in `misvariable`.

To call `mgee2v`, we issue the command,

```
mgee2v(formula, id, data, corstr="exchangeable", misvariable, valid.sample.ind,
               y.mcformula, x.mcformula, maxit=50, tol=1e-3)
```

where the arguments are described as follows:

- `formula`: a formula object which specifies the relationship between the response and covariates for the observed data.
- `id`: a character object which records individual id in the data.
- `data`: a dataframe or matrix object for the observed data set.
- `corstr`: a character object. The default value is "exchangeable", corresponding to the structure where the association between two paired responses is considered to be a constant. The other option is "log-linear" which indicates the log-linear association between two paired responses.
- `misvariable`: a character object which names the error-prone covariate W.
- `valid.sample.ind`: a string object which names the indicator variable delta. When a data point belongs to the validation set, delta = 1; otherwise 0.
- `y.mcformula`: a string object which indicates the misclassification formula between true response `Y` and the surrogate response S.
- `x.mcformula`: a string object which indicates the misclassification formula between true error-prone covariate X and the surrogate W.
- `maxit`: an integer which specifies the maximum number of iterations. The default is 50.
- `tol`: a numeric object which indicates the tolerance threshold. The default is 1e-3.

The function `mgee2v` returns a list of components:

- `beta`: the coefficients in the same order as that specified in the formula for the response and covariates.
- `alpha`: the coefficients for paired responses global odds ratios. The number of α coefficients corresponds to the paired responses odds ratio structure selected in `corstr`. When `corstr="exchangeable"`, only one baseline α is fitted.

- `variance`: the variance-covariance matrix of the estimators of all parameters.
- `convergence`: a logical variable; TRUE if the model converges.
- `iteration`: the number of iterations for the estimates of the model parameters to converge.
- `call`: an unevaluated function call which consists of the named function applied to the given arguments.

ordGEE2

In addition to developing the package *mgee2* to implement the methods of Chen et al. (2014), which accommodate misclassification effects in inferential procedures, we also implement the naive method of ignoring the feature of misclassification and call the resulting function `ordGEE2`. This function can be used together with the preceding described `mgee2k` or `mgee2v` to evaluate the impact of not addressing misclassification effects:

```
ordGEE2(formula, id, data, corstr = "exchangeable", maxit = 50, tol = 0.001)
```

Data analysis

In this section, we conduct numerical studies to demonstrate the usage of our developed R package as well as to show supplementary functions such as summary and plot functions in this package. We first demonstrate all of the external functions in `mgee2` through an example with a simulated data set, known as `obs1`, provided in our package.

An example

The simulated data set, called "`obs1`", includes 8 columns and 3000 rows, with each patient having 3 entries of visits. The format of this data set is as follows.

```
> head(obs1)
ID    Y    X treatment visit S W delta
1  1    2    2         1     1 2 2     1
2  1    0    0         1     2 0 0     1
3  1 <NA> <NA>         1     3 1 2     0
4  2 <NA> <NA>         1     1 1 0     0
5  2 <NA> <NA>         1     2 0 1     0
6  2 <NA> <NA>         1     3 0 0     0
> summary(obs1)
ID             Y           X        treatment visit
Min.   :   1.0   0   : 352   0   : 444   0:1500    1:1000
1st Qu.: 250.8   1   : 283   1   : 269   1:1500    2:1000
Median : 500.5   2   : 256   2   : 178             3:1000
Mean   : 500.5   NA's:2109   NA's:2109
3rd Qu.: 750.2
Max.   :1000.0
S        W           delta
0:1181   0:1460   Min.   :0.000
1: 955   1: 944   1st Qu.:0.000
2: 864   2: 596   Median :0.000
Mean   :0.297
3rd Qu.:1.000
Max.   :1.000
```

Here, `Y` and X represent the true outcome and covariate variables, both being ordinal variables, each taking 3 possible values, denoted 0, 1, and 2, whereas S and W are the observed surrogates for `Y` and X, respectively, with a 5% misclassification rate. `delta` is 1 when the subject is in the validation set and 0 otherwise. About 30% of subjects are randomly chosen to be included in the validation set. We include the subscripts i and j to `Y` and X to indicate the measurements for the corresponding variables for subject i at time point j, in considering the proportional odds model indicated by (1),

$$\text{logit}\, \lambda_{ijk} = \beta_{0k} + \beta_{X1}X_{ij1} + \beta_{X2}X_{ij2} + \beta_{Z1}Z_{ij1} + \beta_{Z2}Z_{ij2} + \beta_{Z3}Z_{ij3} \text{ for } k = 1, 2,$$

where λ_{ijk} is defined as for (1); the treatment variable, denoted Z_{ij1}, is an error-free binary variable; we simulated 3 visits for each patient, denoted by dummy variables Z_{ij2} and Z_{ij3}, with the first visit as a reference level; X_{ij1} and X_{ij2} represent the dummy variables to reflect the three levels of the error-prone covariate, X_{ij}; and $(\beta_{0k}, \beta_{x1}, \beta_{x2}, \beta_{z1}, \beta_{z2}, \beta_{z3})^T$ is the vector of regression coefficients.

In the case corstr = "exchangeable", the association, defined as in (2), between paired responses is assumed to be

$$\log \psi_{i,jk,j'k'} = \phi;$$

while in the case corstr = "log-linear", the association is assumed to be

$$\log \psi_{i,jk,j'k'} = \phi + \phi_2 I(k=2) + \phi_2 I(k'=2) + \phi_{22} I(k=2, k'=2),$$

where ϕ, ϕ_2, and ϕ_{22} are parameters.

We now apply mgee2k and mgee2v, in contrast to ordGEE2, to fit the data to the models, respectively. The results are displayed as follows. In the summary tables for the R output, we use "Y>=1" and "Y>=2" to denote the coefficients β_{01} and β_{02}, respectively, and let "Delta" correspond to the parameter ϕ in the dependence structure.

mgee2k

To use function mgee2k, we need to specify the misclassification matrices beforehand. Here, we set the misclassification matrices the same as used in the simulation process.

```
> data(obs1)
> obs1$visit <- as.factor(obs1$visit)
> obs1$treatment <- as.factor(obs1$treatment)
> obs1$S <- as.factor(obs1$S)
> obs1$W <- as.factor(obs1$W)
> ## set misclassification parameters to be known.
> varphiMat <- gamMat <- log( cbind(0.04/0.95, 0.01/0.95,
+                                   0.95/0.03, 0.02/0.03,
+                                   0.04/0.01, 0.95/0.01) )
> mgee2k.fit = mgee2k(formula = S~W+treatment+visit, id = "ID", data = obs1,
+                    corstr = "exchangeable", misvariable = "W", gamMat = gamMat,
+                    varphiMat = varphiMat)
> summary(mgee2k.fit)
Call:
mgee2k(formula = S ~ W + treatment + visit, id = "ID", data = obs1,
corstr = "exchangeable", misvariable = "W", gamMat = gamMat,
varphiMat = varphiMat)

Summary table of the estimation
            Estimate  Std.Err Z value   Pr(>z)
Y>=1         0.70889  0.08591   8.251 2.22e-16 ***
Y>=2        -0.67521  0.08625  -7.828 4.88e-15 ***
W1           0.58667  0.08719   6.729 1.71e-11 ***
W2           0.94948  0.09745   9.743  < 2e-16 ***
treatment1 -0.70554  0.09114  -7.742 9.77e-15 ***
visit2      -0.24147  0.07735  -3.122   0.0018 **
visit3      -0.62480  0.07571  -8.253 2.22e-16 ***
Delta        1.22606  0.12231  10.024  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

mgee2v

To use mgee2v, a column of indicator variable should be specified in valid.sample.ind.

```
> data(obs1)
>   obs1$visit <- as.factor(obs1$visit)
>   obs1$treatment <- as.factor(obs1$treatment)
>   obs1$S <- as.factor(obs1$S)
>   obs1$W <- as.factor(obs1$W)
>   mgee2v.fit = mgee2v(formula = S~W+treatment+visit, id = "ID", data = obs1,
+                       y.mcformula = "S~1", x.mcformula = "W~1",
+                       misvariable = "W", valid.sample.ind = "delta",
+                       corstr = "exchangeable")
> summary(mgee2v.fit)
Call:
mgee2v(formula = S ~ W + treatment + visit, id = "ID",
```

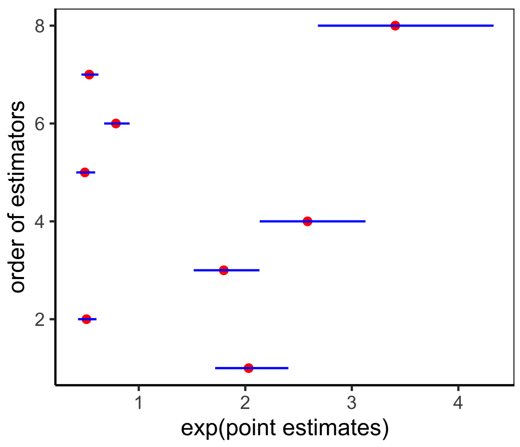


Figure 1: The display of the results in the summary table of applying mgee2k method to the example. The vertical axis presents the estimates for the coefficients corresponding to Y>=1, Y>=2, . . . , and Delta in the order from the bottom to the top. The horizontal axis shows exp(point estimates) (shown in red dots) for those coefficients indicated by the vertical axis, together with their 95% confidence intervals (shown in blue line segments). The confidence intervals are calculated as $(\exp(C_L), \exp(C_U))$, where (C_L, C_U) is a 95% confidence interval of a coefficient.

```
data = obs1, corstr = "exchangeable", misvariable = "W",
valid.sample.ind = "delta", y.mcformula = "S~1", x.mcformula = "W~1")

Summary table of the estimation
            Estimate  Std.Err Z value   Pr(>z)
Y>=1         0.64876  0.08851   7.330 2.30e-13 ***
Y>=2        -0.68226  0.08703  -7.839 4.44e-15 ***
W1           0.56507  0.08140   6.942 3.88e-12 ***
W2           0.98411  0.09305  10.577  < 2e-16 ***
treatment1 -0.68153  0.09052  -7.529 5.11e-14 ***
visit2     -0.24694  0.07483  -3.300 0.000966 ***
visit3     -0.60027  0.07335  -8.184 2.22e-16 ***
Delta        1.22862  0.12160  10.103  < 2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

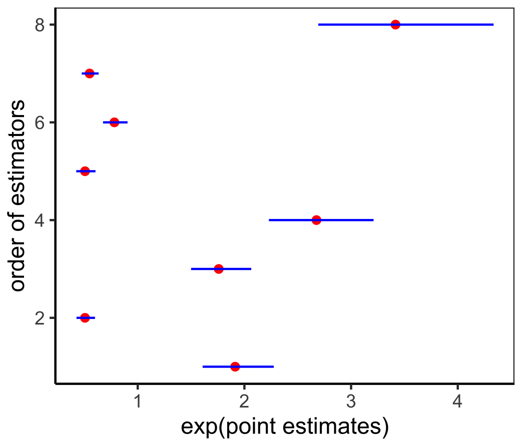


Figure 2: The display of the results in the summary table of applying mgee2v method to the example. The vertical axis presents the estimates for the coefficients corresponding to Y>=1, Y>=2, . . . , and Delta in the order from the bottom to the top. The horizontal axis shows exp(point estimates) (shown in red dots) for those coefficients indicated by the vertical axis, together with their 95% confidence intervals (shown in blue line segments). The confidence intervals are calculated as $(\exp(C_L), \exp(C_U))$, where (C_L, C_U) is a 95% confidence interval of a coefficient.

ordGEE2

```
> naigee.fit = ordGEE2(formula = S~W+treatment+visit, id = "ID",
+                       data = obs1, corstr = "exchangeable")
> summary(naigee.fit)
Call:
ordGEE2(formula = S ~ W + treatment + visit, id = "ID",
data = obs1, corstr = "exchangeable")

Summary table of the estimation
            Estimate  Std.Err Z value   Pr(>z)
Y>=1         0.73276  0.07990   9.171  < 2e-16 ***
Y>=2        -0.69330  0.08004  -8.662  < 2e-16 ***
W1           0.51237  0.07354   6.967 3.23e-12 ***
W2           0.84890  0.08582   9.892  < 2e-16 ***
treatment1 -0.65954  0.08511  -7.749 9.33e-15 ***
visit2     -0.22766  0.07241  -3.144  0.00167 **
visit3     -0.58407  0.07052  -8.282 2.22e-16 ***
Delta       1.06616  0.09846  10.828  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1
```

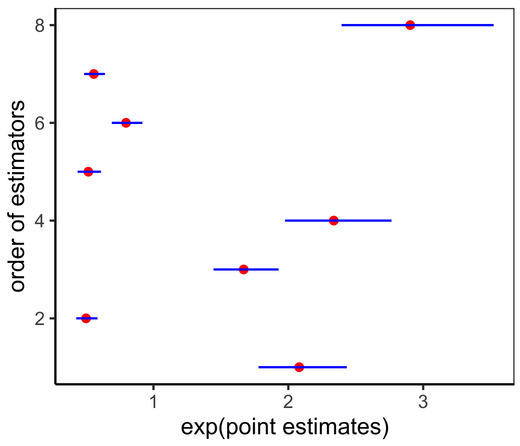


Figure 3: The display of the results in the summary table of applying ordGEE2 method to the example. The vertical axis presents the estimates for the coefficients corresponding to Y>=1, Y>=2, ..., and Delta in the order from the bottom to the top. The horizontal axis shows exp(point estimates) (shown in red dots) for those coefficients indicated by the vertical axis, together with their 95% confidence intervals (shown in blue line segments). The confidence intervals are calculated as $(\exp(C_L), \exp(C_U))$, where (C_L, C_U) is a 95% confidence interval of a coefficient.

plot_model

We use the function plot_model to compare the results obtained from the three functions:

```
> plot_model(naigee.fit)
> plot_model(mgee2.fit)
> plot_model(mgee2v.fit)
```

It is helpful to compare the odds ratios when there are multiple covariates. We use the function plot_model to visualize the odds ratios. The estimated odds ratios for this simulated data set across the three methods are displayed in Figure 1, 2, and 3. The red dot gives the odds ratio of each covariate. The horizontal blue line measures the length of each confidence interval. The vertical axes of the graphs indicate the descending order of the covariates. In other words, the red points from the lowest to the highest in the graph represent the first covariate, the second covariate, and so on. It is seen that the three methods yield similar odds ratios.

Simulation studies

To further compare the three methods, a simulation study is conducted. We run 500 simulations where each data set includes 1000 subjects, with three visits for each subjects. obs1 is one example of the simulated data. The true values of the coefficients are reported in Table 1:

β_{01}	β_{02}	β_{X1}	β_{X2}	β_{Z1}	β_{Z2}	β_{Z3}
$\log 2$	$\log(1/2)$	$\log 2$	$\log 3$	$\log(1/2)$	$\log(3/4)$	$\log(1/2)$

Table 1: True coefficients.

Table 2 reports the simulation results for the care with a 5% misclassification rate set for both the response and covariate variables, where Bias% records a bias in percentage, EV represents an empirical variance, AMV stands for an average of model-based variance, and CR records a coverage rate of 95% confidence intervals. Simulation results show that the `mgee2` and `mgee2v` perform better than the naive method `ordGEE2`, and they produce reasonable results.

	ordGEE2				mgee2				mgee2v			
	Bias%	EV	AMV	CR	Bias%	EV	AMV	CR	Bias%	EV	AMV	CR
β_{01}	3.119	0.007	0.007	0.942	-0.915	0.008	0.008	0.944	-2.223	0.008	0.008	0.951
β_{02}	3.226	0.007	0.007	0.946	1.562	0.008	0.008	0.940	3.556	0.009	0.008	0.947
β_{x1}	-12.112	0.006	0.006	0.784	1.238	0.008	0.008	0.942	-6.933	0.024	0.014	0.924
β_{x2}	-9.810	0.007	0.008	0.754	1.433	0.009	0.010	0.964	3.393	0.016	0.011	0.941
β_{z1}	-7.032	0.008	0.007	0.922	-0.456	0.009	0.008	0.954	-0.138	0.009	0.008	0.949
β_{z2}	-6.311	0.006	0.005	0.932	0.071	0.006	0.006	0.938	-0.364	0.006	0.006	0.932
β_{z3}	-6.630	0.005	0.005	0.908	0.056	0.006	0.006	0.964	0.143	0.006	0.006	0.962
ϕ	-13.130	0.009	0.010	0.690	0.217	0.014	0.015	0.954	1.257	0.016	0.017	0.956

Table 2: Simulation results with a 5% misclassification rate.

In addition to the preceding simulation with a misclassification rate of 5%, we conducted another simulation with the same parameters except that the misclassification rate is changed to be 20%, and `corstr = "log-linear"`. The results are reported in Table 3, which shows more noticeable differences in implementing the three functions, 'ordGEE2', 'mgee2', and 'mgee2v'.

	ordGEE2				mgee2				mgee2v			
	Bias%	EV	AMV	CR	Bias%	EV	AMV	CR	Bias%	EV	AMV	CR
β_{01}	9.589	0.007	0.007	0.866	0.748	0.015	0.015	0.952	0.872	0.015	0.016	0.966
β_{02}	11.131	0.006	0.007	0.842	-0.210	0.014	0.015	0.958	-0.523	0.015	0.016	0.958
β_{x1}	-48.891	0.005	0.006	0.000	-1.233	0.027	0.029	0.964	-0.874	0.023	0.023	0.940
β_{x2}	-43.506	0.008	0.008	0.000	-0.400	0.026	0.027	0.958	-0.399	0.023	0.023	0.946
β_{z1}	-26.284	0.006	0.006	0.346	0.364	0.011	0.012	0.960	0.084	0.011	0.011	0.940
β_{z2}	-24.870	0.006	0.006	0.832	1.703	0.012	0.011	0.938	1.396	0.010	0.010	0.948
β_{z3}	-26.893	0.006	0.006	0.314	0.228	0.011	0.012	0.954	0.144	0.010	0.011	0.968
ϕ_0	-53.210	0.009	0.009	0.000	1.873	0.078	0.068	0.942	1.034	0.052	0.066	0.976
ϕ_2	-73.139	0.006	0.006	0.042	-0.353	0.052	0.047	0.948	-1.241	0.037	0.049	0.970
ϕ_{22}	-59.955	0.011	0.011	0.000	0.256	0.075	0.075	0.944	0.188	0.053	0.079	0.978

Table 3: Simulation results with a 20% misclassification rate.

A case study

To illustrate the usage of the developed R package, we analyze a dataset arising from the Framingham Heart Study, obtained from the NIH website (https://biolincc.nhlbi.nih.gov/teaching/). Similar to Chen et al. (2014), we consider those 915 male patients who completed both exams #2 and #3, and age between 31 and 65 at the entry of the study. The response variable, HBP, is a categorical variable indicating the status of the systolic blood pressure (SBP), where HBP=0 if SBP is below 140 mmHg, HBP=1 if SBP is between 140 mmHg and 159 mmHg, and HBP=2 if SBP is larger than 160 mmHg.

We are interested in understanding the relationship between HBP and covariates, including the serum cholesterol level (CHOL), age, and the current smoking status (CURSMOKE), as well as the

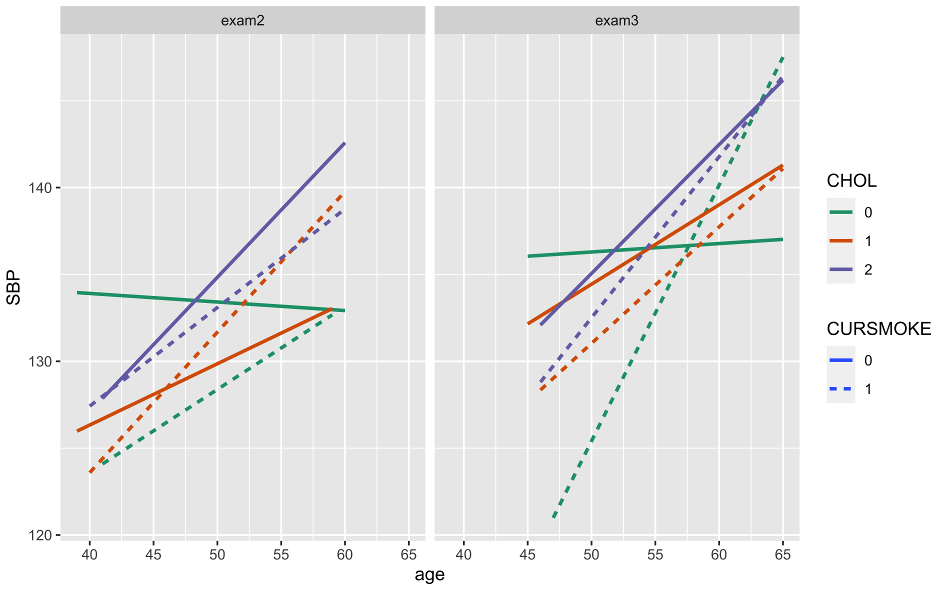


Figure 4: The least squares regression lines by fitting scattered data of SBP against age under different categories stratified by the combination of current smoking status (CURSMOKE) and cholesterol level(CHOL), for patients at different exam times. For example, the dotted red line on the left panel is a linear model fit for SBP against AGE for smokers with level 1 cholesterol at exam 2.

examination status, denoted as "Exam3". CHOL is classified as three categories, with 0, 1, and 2 representing normal (less than 200 mg/dL), borderline high (200-239mg/dL), and hypercholesterolemia (greater than 240 mg/dL), respectively. Exam3 is a dummy variable, with 1 indicating observations for exam 3 and 0 for exam 2.

First, we visualize how SBP may change with age by stratifying the study subjects into different categories according to the exam time, smoking status, or CHOL. To see the trend, we display simple linear regression lines that fit scattered points of SBP against AGE for patients in each category, as shown in Figure 4. Except for the patients with CHOL value 0 and CURSMOKE value 0 at exam 2, there is generally an upward trend of SBP versus age for each category, though the degree varies. While each patient takes 2 exams, the time interval between two exams is different from patient to patient. To reflect this feature of different gap times for the study subjects, in Figure 5 we further display spaghetti plots (Hedeker and Gibbons, 2006) for patients in different categories, where the two endpoints of each black line segment mark SBP and age for the corresponding study subject at exams 2 and 3 in each category, respectively. The blue curve represents the loess smooth curve in each panel to show the trend of SBP against AGE. The loess smooth function is a tool to create smooth lines for scattered plots using polynomial approximations. The code for producing Figures 4 and 5 is included in the help file of data set *heart* in our R package.

Next, we use the proportional odds model to examine how SBP may be quantitatively associated with the covariates. For the ith patient at the jth visit, $X_{ij,CHOL=1}$ and $X_{ij,CHOL=2}$ are binary indicator variables recording whether the patient's cholesterol level is 1 and 2, respectively; $Z_{ij,smoker}$ is a binary variable whether or not the patient is a smoker; $Z_{ij,exam3}$ is a binary variable showing whether or not the patient is taking exam #3; and $Z_{i,age}$ records the age of the ith patient at the entry of the study.

As defined in (1), consider the model

$$\begin{aligned}\text{logit}\,\lambda_{ijk} =&\beta_{0k} + \beta_{X,CHOL=1}X_{ij,CHOL=1} + \beta_{X,CHOL=2}X_{ij,CHOL=2}\\ &+ \beta_{Z,age}Z_{i,age} + \beta_{Z,smoker}Z_{ij,smoker} + \beta_{Z,exam3}Z_{ij,exam3}\end{aligned} \tag{8}$$

for $k = 1, 2$, where $\beta_{0k}, \beta_{X,CHOL=1}, \beta_{X,CHOL=2}, \beta_{Z,age}, \beta_{Z,smoker}$, and $\beta_{Z,exam3}$ are the parameters.

The data set used in our example is included in our package called "`heart`". To demonstrate the usage of the developed package, we perceive that the response HBP level and the covariate cholesterol level are prone to misclassification. Since this example does not have a validation data set, we only analyze the data using the naive method, "`ordGEE2`", and the corrected method with a specified known misclassification rate, "`mgee2k`", where the misclassification rates for both the outcome and the covariate are assumed to be 5%, and the exchangeable dependence structure is considered. The analysis results are shown in Table 4. Overall, the naive method and the corrected method indicate the same significant health factors, yet the magnitude of the coefficient estimates and their standard

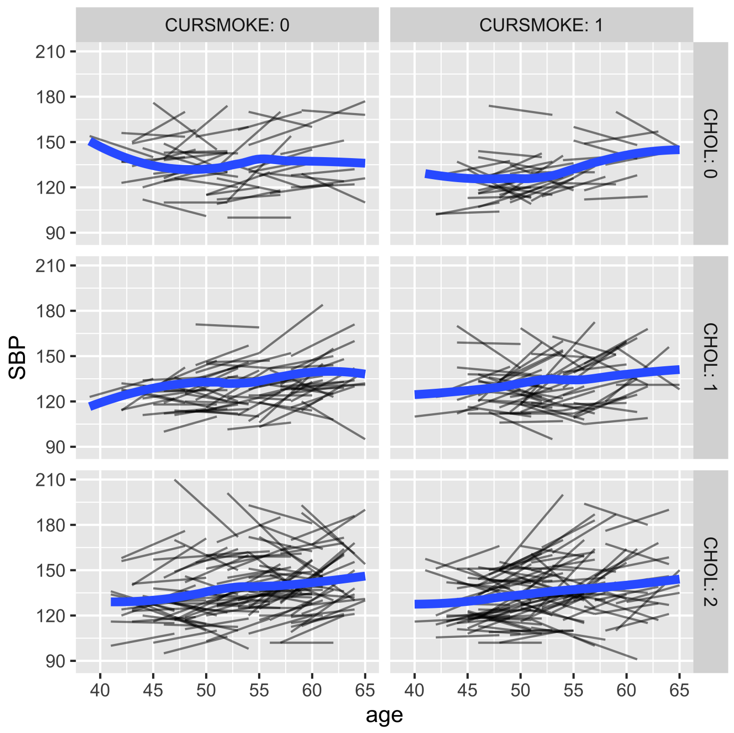


Figure 5: The spaghetti plots of SBP at exams 2 and 3 for the patients classified into different groups by different values of current smoking status and cholesterol level, where the varying lengths of the black line segments reflect the fact that the gap time between exams 2 and 3 differ from patient to patient. The blue curve in each panel is fitted using the loess function.

errors are different. Higher cholesterol levels and older ages appear to be positively correlated with high blood pressure.

Summary

Analysis of longitudinal ordinal data is important for research in health science, epidemiological studies, and social science. Marginal analysis using generalized estimating equations has been extensively employed in applications. However, such a strategy is challenged by the presence of mismeasurement of variables. To address this challenge, Chen et al. (2014) developed estimation methods for analyzing correlated ordinal responses and ordinal covariates, which are subject to misclassification.

To allow analysts to apply the useful methods of Chen et al. (2014) without doing individual codes, we develop an R package *mgee2* to implement the methods for general use. Our package provides three methods for estimation, including the two methods of corrections for misclassification effects, as opposed to the naive method, which disregards the feature of mismeasurement in variables. The package can be used for modeling longitudinal ordinal data with misclassified response and covariates. It provides consistent estimation results by directly inputting the data under required assumptions.

Acknowledgements

The authors thank the review team for the helpful comments on the initial version. Yi's research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Yi is Canada Research Chair in Data Science (Tier 1). Her research was undertaken, in part, thanks to funding from the Canada Research Chairs program.

	ordGEE2			mgee2k		
	Est.	SD	p-vlue	Est.	SD	p-vlue
β_{01}	-4.291	0.635	<0.001	-4.943	0.737	<0.001
β_{02}	-5.623	0.638	<0.001	-6.195	0.740	<0.001
$\beta_{x,CHOL=1}$	0.068	0.133	0.608	0.117	0.180	0.515
$\beta_{x,CHOL=2}$	0.352	0.140	0.012	0.474	0.186	0.011
$\beta_{z,age}$	0.063	0.012	<0.001	0.071	0.014	<0.001
$\beta_{z,smoker}$	-0.044	0.105	0.673	-0.042	0.118	0.722
$\beta_{z,exam3}$	0.145	0.097	0.133	0.182	0.110	0.097
ϕ	2.301	0.207	<0.001	2.301	0.318	<0.001

Table 4: A case study of a data subset arising from the Framingham Heart Study, with an assumed misclassification rate at 5%.

Bibliography

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2019. URL `https://CRAN.R-project.org/package=Matrix`. R package version 1.2-17. [p474]

L. J. Beesley and B. Mukherjee. Statistical inference for association studies using electronic health records: handling both selection bias and outcome misclassification. *Biometrics*, 2020. [p471]

V. J. Carey. *gee: Generalized Estimation Equation Solver*, 2015. URL `https://CRAN.R-project.org/package=gee`. R package version 4.13-19. [p471]

R. J. Carroll, D. Ruppert, L. A. Stefanski, and C. M. Crainiceanu. *Measurement Error in Nonlinear Models*. Chapman and Hall/CRC, 2006. URL `https://doi.org/10.1201/9781420010138`. [p471]

Z. Chen, G. Y. Yi, and C. Wu. Marginal methods for correlated binary data with misclassified responses. *Biometrika*, 98(3):647–662, 2011. URL `http://www.jstor.org/stable/23076137`. [p471]

Z. Chen, G. Y. Yi, and C. Wu. Marginal analysis of longitudinal ordinal data with misclassification in both response and covariates. *Biometrical Journal*, 56(1):69–85, 2014. URL `https://doi.org/10.1002/bimj.201200195`. [p471, 472, 473, 474, 476, 480, 482]

S. Dlugosz, E. Mammen, and R. A. Wilke. Generalized partially linear regression with misclassified data and an application to labour market transitions. *Computational Statistics & Data Analysis*, 110: 145–159, 2017. [p471]

C. Genolini, X. Alacoque, M. Sentenac, and C. Arnaud. kmlandkml3d: R packages to cluster longitudinal data. *Journal of Statistical Software*, 65(4), 2015. URL `https://doi.org/10.18637/jss.v065.i04`. [p471]

D. Hedeker and R. D. Gibbons. *Longitudinal Data Analysis*. John Wiley & Sons, 2006. [p481]

K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73 (1):13–22, 1986. URL `https://doi.org/10.1093/biomet/73.1.13`. [p471]

J. M. Neuhaus. Bias and efficiency loss due to misclassified responses in binary regression. *Biometrika*, 86(4):843–855, 1999. URL `http://www.jstor.org/stable/2673589`. [p471]

M. S. Pepe and G. L. Anderson. A cautionary note on inference for marginal regression models with longitudinal data and general correlated response data. *Communications in Statistics - Simulation and Computation*, 23(4):939–951, 1994. URL `https://doi.org/10.1080/03610919408813210`. [p472]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. [p474]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL `https://ggplot2.tidyverse.org`. [p474]

J. M. Williamson, K. Kim, and S. R. Lipsitz. Analyzing bivariate ordinal data using a global odds ratio. *Journal of the American Statistical Association*, 90(432):1432–1437, 1995. URL `https://doi.org/10.1080/01621459.1995.10476649`. [p472]

J. Xiong and G. Y. Yi. swgee: An r package for analyzing longitudinal data with response missingness and covariate measurement error. *The R Journal*, 11(1):416–426, 2019. [p471]

C. Xu, Z. Li, and M. Wang. *wgeesel: Weighted Generalized Estimating Equations and Model Selection*, 2018. URL `https://CRAN.R-project.org/package=wgeesel`. R package version 1.5. [p471]

G. Y. Yi. A simulation-based marginal method for longitudinal data with dropout and mismeasured covariates. *Biostatistics*, 9(3):501–512, 2008. URL `https://doi.org/10.1093/biostatistics/kxm054`. [p471]

G. Y. Yi. *Statistical Analysis with Measurement Error or Misclassification: Strategy, Method and Application*. Springer, New York, 2017. [p471]

G. Y. Yi, Y. Ma, D. Spiegelman, and R. J. Carroll. Functional and structural methods with mixed measurement error and misclassification in covariates. *Journal of the American Statistical Association*, 110(510):681–696, 2015. [p471]

Q. Zhang and G. Y. Yi. R package for analysis of data with mixed measurement error and misclassification in covariates: augsimex. *Journal of Statistical Computation and Simulation*, 89(12):2293–2315, 2019. [p471]

Yuliang Xu
School of Public Health
University of Michigan
500 S. State Street, Ann Arbor, Michigan 48109 USA
`yuliangx@umich.edu`

Shuo Shuo Liu
Department of Statistics
Pennsylvania State University
201 Old Main, University Park, Pennsylvania 16802 USA
ORCiD: 0000-0001-8396-4515
`shuoshuo.liu@psu.edu`

Grace Y. Yi
Department of Statistical and Actuarial Sciences
Department of Computer Science
University of Western Ontario
1151 Richmond Street, London, Ontario, Canada N6A 3K7
`gyi5@uwo.ca`

A New Versatile Discrete Distribution

by Rolf Turner

Abstract This paper introduces a new flexible distribution for discrete data. Approximate moment estimators of the parameters of the distribution, to be used as starting values for numerical optimization procedures, are discussed. "Exact" moment estimation, effected via a numerical procedure, and maximum likelihood estimation, are considered. The quality of the results produced by these estimators is assessed via simulation experiments. Several examples are given of fitting instances of the new distribution to real and simulated data. It is noted that the new distribution is a member of the exponential family. Expressions for the gradient and Hessian of the log-likelihood of the new distribution are derived. The former facilitates the numerical maximization of the likelihood with `optim()`; the latter provides means of calculating or estimating the covariance matrix of of the parameter estimates. A discrepancy between estimates of the covariance matrix obtained by inverting the Hessian and those obtained by Monte Carlo methods is discussed.

Introduction

Modelling the distribution of discrete data sets can be problematic in that it is often the case that none of the "standard" distributions appears to be appropriate. It is possible to use a "completely nonparametric" approach (in other words, to apply multinomial distributions, specified in a very simple manner, by means of tables). However, this approach often turns out to be a little *too* flexible. In particular, in the context of hidden Markov models for discrete data (**hmm.discnp**, Turner 2020), the number of quantities to estimate rapidly becomes unwieldy. Estimates are unstable, the sensitivity of fitting algorithms to starting values is exacerbated, and problems with the convergence of fitting algorithms arise.

To address these problems, I developed a new discrete distribution, termed the "db" ("discretized Beta") distribution. The underlying idea is to define a family of distributions, for discrete data, with shape characteristics as flexible as those of the Beta family of continuous distributions. (See Johnson et al. 1995, Chapter 25, p. 210. See also the help for the `dbeta()` function in the **stats** package, R Core Team 2020, and Abramowitz and Stegun 1972, Chapter 6. The reader may also find it useful to access `https://en.wikipedia.org/wiki/Beta_distribution`.) The db distribution is closely related to the Beta distribution and has "shape" parameters, α and β, analogous to the shape parameters of the Beta distribution.

In addition to the shape parameters, the db distribution has two other parameters which specify the "support" of the distribution. These "support parameters" are not estimated from data but must be specified by the user prior to estimating the shape parameters. The support parameters are n_{top} (a positive integer) and ζ (a logical scalar).

The parameter n_{top} is the upper limit of the support of the specified distribution. If the parameter ζ is `TRUE` then zero origin indexing is to be used, in which case the support of the distribution is the set $\{0, 1, 2, \ldots, n_{\text{top}}\}$. Otherwise the support is $\{1, 2, \ldots, n_{\text{top}}\}$. In the first case I use the notation $n_{\text{bot}} = 0$ and in the second $n_{\text{bot}} = 1$. The first form is convenient if the variable in question may be considered to be a count and zero counts are possible. Of course, one could structure the distribution always to have support of the form $\{0, 1, 2, \ldots, n_{\text{top}}\}$, simply by re-coding or shifting the data. However, in several of the examples with which I was concerned, it seemed more convenient to allow for a non-zero origin.

In some contexts the value of n_{top} may be known (e.g., it may be analogous to the number of trials in a binomial experiment). In other contexts it must be chosen by the user, and the choice may be influenced by the observed values of the data. (See the section **Choosing n_{top}**.)

Like the Beta distribution upon which it is based, the db distribution is effectively unimodal. It can have two modes if they occur at the extremes of the support but otherwise can have only one. This characteristic is less than ideal, but seems to be unavoidable. It appears to be difficult to specify multimodal distributions (other than by way of *mixtures*, which are accompanied by other problems). *Wikipedia* (`https://en.wikipedia.org/wiki/Multimodal_distribution`, last accessed 30 March 2021) says *"Bimodal distributions, despite their frequent occurrence in data sets, have only rarely been studied [citation needed]* (sic). *This may be because of the difficulties in estimating their parameters either with frequentist or Bayesian methods."*

A referee of an earlier version of this paper suggested that the beta-binomial distribution be considered as an alternative to the new db distribution. This referee pointed out to me the paper "Modeling the patient mix for risk-adjusted CUSUM charts" by Philipp Wittenberg, which has interesting applications in medical science. In this paper, which is to appear in *Statistical Methods in Medical*

Research, the beta-binomial distribution is used to model the distribution underlying a large data set of integer-based Parsonnet risk scores (see Parsonnet et al. 1989; see also Steiner et al. 2000). In addition to modeling the Parsonnet risk scores with the beta-binomial distribution, Wittenberg rescales these scores to lie between 0 and 1 and applies the Beta distribution.

The data in question are available (in a slightly modified form) in the **spcadjust** package (Gandy and Kvaloy 2013). I fitted both a db distribution and a beta-binomial distribution to these data and conducted a goodness of fit test in both instances. The tests indicate that neither distribution is actually appropriate. Both Monte Carlo p-values were 0.01. More detail is given in **Example 6** in section **Examples**.

The beta-binomial distribution was introduced in Skellam (1948). Like the db distribution, it has flexibility of shape similar to that of the Beta distribution. However I have a couple of reservations about this distribution which I discuss in the following section.

The beta-binomial distribution

My first reservation about the beta-binomial distribution is that it is to a large extent focussed on dealing with data which appear to arise from binomial distributions but which are, in fact, "*overdispersed*". In other words, the focus is on data which exhibit extra-binomial variation. Such data have variance which is larger than it would be if the underlying distribution were indeed binomial.

The focus of the beta-binomial distribution on overdispersion would appear to make its application to underdispersed data problematic. Underdispersed pseudo-binomial data sets (i.e. data sets which have variance *smaller* than they would have if the underlying distribution were binomial) are rare, but they do exist. Examples are provided in the **dbd** package (see section **Implementation**). It is indeed possible to fit beta-binomial distributions to these examples, and goodness of fit tests indicate that the fits are adequate. However, the meaning of the resulting fits is questionable. The estimates of s range from around 640 thousand to 78 million. Such large values of s essentially indicate that there is *no* overdispersion, i.e., that the data are, in fact, from a binomial distribution. In other words, the estimates are trying as hard as they can to describe the true situation but cannot actually do so given the constraints of the model.

In fairness, it must be pointed out that the db distribution does not perform particularly well when applied to the data sets referred to above. There are no obvious theoretical problems with fitting the db distribution to underdispersed data. However, goodness of fit tests reject the adequacy of the fit of the db distribution to one of these data sets. In contrast, the beta-binomial distribution appears to fit this data set adequately. Further details are given in **Example 7** in section **Examples**. The parameter estimates for the rejected fit of the db distribution appear to be excessively large, which might well raise suspicions. It is not clear how the values of parameter estimates obtained from the db distribution relate to under and overdispersion. This may be a topic to explore in future research.

My second reservation about the beta-binomial distribution is that (as revealed by fairly extensive simulation experiments) parameter estimation for this distribution can, from time to time, be unstable. The beta-binomial distribution may be conveniently parameterized in terms of a "success probability" m (which must be strictly between 0 and 1) and an overdispersion parameter s (which must be strictly positive). This is the parameterization chosen in the **rmutil** package (Swihart and Lindsey 2020). The reader may also find it informative to access `https://en.wikipedia.org/wiki/Beta-binomial_distribution`, where the parameterization is expressed in terms of "shape" parameters α and β. These are related to m and s by $m = \alpha/(\alpha + \beta)$ and $s = \alpha + \beta$.

Moment estimators of the parameters of a beta-binomial distribution are explicitly available. However, these are often unsatisfactory in that the moment estimate of s can turn out to be negative. Maximum likelihood estimates of the parameters may be obtained via numerical maximization (using, e.g., `optim()` from the **stats** package, automatically available in R). Starting values are, of course, required. If the moment estimates are outside of the required range, e.g., if $\hat{s}$ is negative, rough ad hoc starting values (e.g., $m = \epsilon$ or $m = 1 - \epsilon$, and $s = \epsilon$, where ϵ is equal to `sqrt(.Machine$double.eps)`) appear to be adequate most of the time.

However, irrespective of starting values, the maximum likelihood estimate of s is frequently far too large. In one instance, I simulated (using `rbetabinom()` from the **rmutil** package) 100 data sets, each with 30 observations, with $m = 0.75$, $s = 10$, and `size` (the number of trials) set equal to 10. Maximum likelihood estimates of s, calculated using the true parameter values as starting values, ranged as high as 1038.82. (An "h" plot of the estimates is shown in Figure 1.) The variance of these estimates was 13032.98. In contrast, the inverse of the Fisher information, calculated using the true parameter values and the data corresponding to the highest estimate of s was

```
            m           s
m 0.001121638  0.02524533
s 0.025245334 14.66327852
```

which indicates that the variance in question should be of the order of 15.

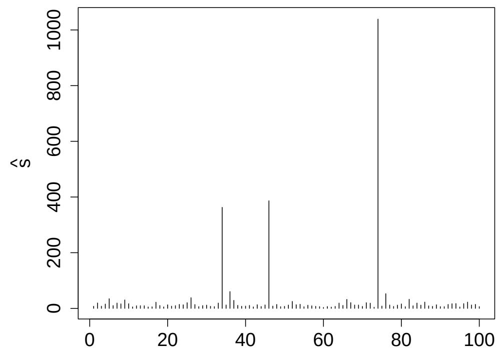


Figure 1: Estimates of the s parameter of the beta-binomial distribution.

I encountered other problems — tendencies for errors to be thrown in various circumstances, including the application of `optim()` — in my exploration of the beta-binomial distribution, but there would appear to be no point in going into more detail here.

Despite these problems and my reservations about the beta-binomial distribution, I have included a complete set of tools for working with this distribution in the **dbd** package. These tools have a structure exactly analogous to that of the tools provided for working with the db distribution.

Definition of the db distribution

Conceptually, the probability mass function (PMF) of the db distribution is

$$\begin{aligned} \Pr(X = x \mid \alpha, \beta, n_{\text{top}}, \zeta) &= \frac{1}{\kappa} f\left(\frac{x - n_{\text{bot}} + 1}{n_{\text{top}} - n_{\text{bot}} + 2}\right), \text{ where} \\ \kappa &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} f\left(\frac{i - n_{\text{bot}} + 1}{n_{\text{top}} - n_{\text{bot}} + 2}\right) \end{aligned} \tag{1}$$

$x = n_{\text{bot}}, n_{\text{bot}} + 1, \ldots, n_{\text{top}}$. In (1), $f(\cdot)$ is the probability density function (pdf) of the Beta distribution with the first shape parameter equal to α and the second shape parameter equal to β. The probabilities given by (1) are the values of the corresponding Beta density, evaluated at equispaced points in the interior of the interval $(0, 1)$, normalized to sum to 1. However, it is possible and advantageous to express the definition of the db distribution in a direct manner without making reference to the Beta distribution.

Deriving the direct expression for the PMF of the db distribution from (1) is facilitated by noting that the Beta distribution is a member of the exponential family. From this, it follows that the db distribution as defined by (1) is, for fixed values of the support parameters n_{top} and ζ, also a member. An expression for the PMF of the db distribution, in exponential family form, is derived from the pdf of the Beta distribution in **Appendix I**.

From this derivation, it is seen that the PMF of the db distribution, given by (1), can also be expressed as

$$\begin{aligned} &\Pr(X = x \mid \alpha, \beta, n_{\text{top}}, \zeta) = h(x) \exp\{\alpha T_1(x) + \beta T_2(x) - A(\alpha, \beta)\}, \\ &x = n_{\text{bot}}, n_{\text{bot}} + 1, \ldots, n_{\text{top}}, \text{ where} \end{aligned} \tag{2}$$

$$\begin{aligned}
h(x) &= \frac{(n_{\text{top}} - n_{\text{bot}} + 2)^2}{(x - n_{\text{bot}} + 1)(n_{\text{top}} - x + 1)} \\
T_1(x) &= \log((x - n_{\text{bot}} + 1)/(n_{\text{top}} - n_{\text{bot}} + 2)) \\
T_2(x) &= \log((n_{\text{top}} - x + 1)/(n_{\text{top}} - n_{\text{bot}} + 2) \text{ and} \\
A(\alpha, \beta) &= \log\left(\sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) \exp\{\alpha T_1(i) + \beta T_2(i)\}\right) .
\end{aligned}$$

Consequently the *definition* of the db distribution may be taken to be (2). This has the following advantage. The expression given by (2) is well-defined for *all* values of α and β: positive, negative or zero, whereas (1) is well-defined only for α and β strictly greater than zero. The difference is due to the fact that the pdf of the Beta distribution involves the expression $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha + \beta)$. This latter expression evaluates to ∞ (or `Inf` in R) if either α or β is less than or equal to zero. Consequently, in such cases, (1) is undefined (being equal to `Inf/Inf = NaN` in R). Algebraically, $B(\alpha, \beta)$ cancels from (1) due to the division by κ, whence it does not appear in (2).

Although the db distribution is well-defined for negative parameter values, there are indications of problems in respect of such values. In practice, it may be advisable to restrict attention to positive values only. There certainly *exist* data sets — as can be demonstrated by simulation — for which the (maximum likelihood) estimates of the parameters are undeniably negative. Cursory experimentation using the **dbd** package indicates that in some instances, negative parameters are reasonably well estimated by maximum likelihood. E.g.,

```
library(dbd)
set.seed(42)
x  <- rdb(100,-2,-3,10)
fx <- mleDb(x,10)
print(as.vector(fx))
```

The preceding code produces estimates $\hat{\alpha} = -2.1645$ and $\hat{\beta} = -3.1365$. Ninety-five percent confidence intervals for α and β (based on the variance entries of the "analytic" covariance matrix — see section **Implementation**) are $[-3.0585, -1.2706]$ and $[-3.9443, -2.3297]$, which contain the true values, equal to -2 and -3, respectively.

On the other hand, there are instances in which the maximum likelihood estimates are very large (either positive or negative) when the true values are relatively small and negative:

```
library(dbd)
set.seed(348)
x  <- rdb(100,alpha=-3,beta=-6,ntop=10,zeta=TRUE)
fx <- mleDb(x,ntop=10,zeta=TRUE,maxit=2000)
print(as.vector(fx))
```

The resulting estimates are $\hat{\alpha} = 73.18$ and $\hat{\beta} = -166.61$ which bear no relation to the "truth".

The only (as far as I can see) real advantage in the fact that the parameter values are permitted to be negative lies in the avoidance of various numerical issues that might otherwise arise in the estimation of parameters of a db distribution. In particular, the legitimacy of negative parameter values removes the necessity of imposing box constraints on the procedure for maximizing the likelihood. It also circumvents difficulties that can otherwise arise in evaluating the Hessian of the log-likelihood when one or both of the parameter estimates is close to zero.

Plots of the probability functions of a number of db distributions are shown in Figure 2. The shapes of these distributions mimic those of the corresponding Beta distributions.

Choosing n_{top}

In fitting a db distribution to an observed data set, it is often sensible to set n_{top} equal to the maximum of the data. On the other hand, if there is a conceptual least upper bound for the support of the distribution (not found amongst the observed values), then one should set n_{top} equal to this conceptual least upper bound. Finally, if the data are conceptually unbounded, then one might wish to set n_{top} equal to the maximum of the observed data $+1$. In this latter case, the value of $\Pr(X = n_{\text{top}})$ might be interpreted as $\Pr(X \geq n_{\text{top}})$.

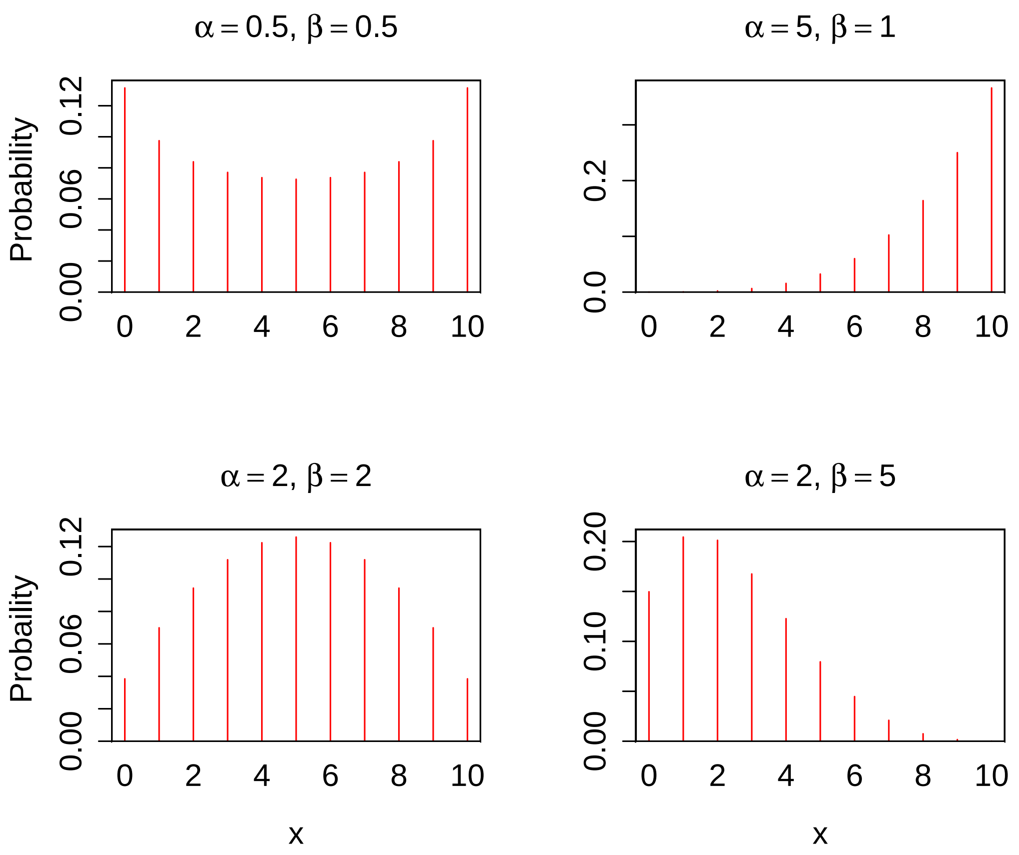


Figure 2: Probability mass functions of db distributions

Implementation

I have written an R package **dbd** (Turner 2021) to provide tools for working with the db distribution. The package supplies the four standard, "d", "p", "q" and "r" — density, probability, quantile and random number generation — functions, that are as a rule associated in R with any given distribution. In this setting, these functions are `ddb()`, `pdb()`, `qdb()`, and `rdb()`. The package also contains functions `expValDb()` and `varDb()` to calculate the expected value (mean) and variance of a db distribution given a specification of its parameters. There is no closed-form algebraic expression for these quantities.

Another useful tool in the package is the function `mleDb()` which enables the easy estimation from data, of the parameters of the db distribution, via maximum likelihood. The function `mleDb()` calls upon an undocumented function `meDb()` which calculates approximate moment estimates of the parameters to serve as starting values for the maximization of the likelihood. A user would not normally make direct use of `meDb()`. However, the approximation used by `meDb()` is of interest in its own right. This approximation is discussed in the section **Estimation of parameters**.

An alternative to `mleDb()` is the function `exactMeDb()`, which is also included in the **dbd** package. This function was suggested by a referee of an earlier version of this paper. It calculates "exact" moment estimates of the parameters by minimizing $(\bar{x} - \mu)^2 + (s^2 - \sigma^2)^2$ where $\bar{x}$ and s^2 are the sample mean and variance, respectively, and μ and σ^2 are the theoretical mean and variance. The latter two quantities are functions of α and β and can be calculated from these parameters using `expValDb()` and `varDb()`. The minimization is accomplished using `optim()`. "Theoretically", the minimum should be zero. The achieved minimum is provided as an attribute `"minSqdiff"` of the value returned by `exactMeDb()` so that the user can see how successful the minimization was. In section **The quality of the estimates**, the "exact" moment estimates are denoted by $\tilde{\alpha}$ and $\tilde{\beta}$.

Somewhat to my surprise, `exactMeDb()` performed (in the simulation experiments that I conducted) essentially as well as `mleDb()`. Occasionally `exactMeDb()` out-performed `mleDdb()` in terms of mean squared error. See section **Quality of the estimators**. On the other hand, it appears that `exactMeDb()` is substantially slower than `mleDb()`. In a small simulation experiment I found that `mleDb()` was about 20 times as fast as `exactMeDb()` (with times measured as the `"user"` time returned by `system.time()`)

when starting values were taken to be the approximate moment estimates, and nearly 30 times as fast when starting values were taken to be the true parameter values.

The function `mleDb()` returns an object of class `"mleDb"`, and there is a corresponding `plot()` method to produce plots of the probability mass functions for distributions having parameters equal to the given estimates. There is also a stand-alone plotting function `plotDb()` which plots the PMF of a db distribution given specified parameters.

The package also has functions that provide means of estimating or calculating the covariance matrix of the parameter estimates. These functions enable the assessment of the uncertainty in the estimates of the parameters. The functions are: `aHess()` ("analytic Hessian"), `nHess()` ("numeric Hessian"), `finfo()`, and `mcCovMat()` (Monte Carlo-based estimate of the covariance matrix). The first three functions provide matrices whose *inverses* are estimates of (or in the case of `finfo()` equal to) the desired covariance matrix, given the supplied parameter values. The function `nHess()` calls upon `optimHess()` from the **stats** package. The function `aHess()` makes use of the expressions set out in **Appendix II**.

The values produced by `aHess()` and `nHess()` generally appear to be in very good agreement. However, if the parameter values are "unreasonably large" (e.g., $\alpha = 150$, $\beta = 400$), then there can be a substantial disparity between the values. In such instances, it would be inadvisable to trust either result.

Of course, the real reason for calculating the Hessian is to obtain an estimate of the covariance matrix of the parameter estimates. Another way to obtain an estimated covariance matrix is to use the Monte Carlo methods conveniently provided by the function `mcCovMat()` referred to above. Again, there is generally good agreement between the value produced by `mcCovMat()` and the inverse of the Hessian produced, e.g., by `aHess()`. However, unless the sample size is quite large, the variance entries of the inverse Hessian are noticeably smaller than those of the matrix returned by `mcCovMat()`:

```
set.seed(25)
x  <- rdb(n=30,alpha=3,beta=4,ntop=10)
fx <- mleDb(x,ntop=10)
solve(aHess(fx))
          alpha     beta
alpha 0.7712032 1.047111
beta  1.0471108 1.790513
 mcCovMat(fx)
         alpha     beta
alpha 1.342672 1.889880
beta  1.889880 3.291645
```

Further discussion of this phenomenon is to be found in **Appendix III**.

The **dbd** package also contains a function `llPlot()` for plotting log-likelihood surfaces and a function `gof()` for performing tests of goodness of fit for the db distribution. The `llPlot()` function may be useful in diagnosing problems with parameter estimation should these arise. The tests effected by `gof()` may be either chi-squared-based tests or Monte Carlo tests. Users should be aware that Monte Carlo tests (which use a relatively small number of simulations) are *random*. Performing a Monte Carlo test is *not* the same as simulating a large number of test statistics (under the null hypothesis) in order to approximate the null distribution of the statistic. See, for example, Baddeley et al. (2015, Section 10.6, p. 384) for some discussion of such tests.

Estimation of parameters

There is, unsurprisingly, no closed-form for any sort of estimates of the shape parameters of a db distribution. Estimates may, however, be calculated reasonably easily via (numerical) maximum likelihood or by solving for moment estimates numerically. The functions `mleDb()` and `exactMeDb()` from the **dbd** package, discussed in the section **Implementation** make use of the `optim()` function from the **stats** package (automatically available in R) to effect the calculations.

The `optim()` function requires starting values for the parameters being estimated. It turns out that adequate starting values can be produced, as indicated in the section **Implementation**, via a (very!) rough explicit approximation to the method of moments. To develop the approximation, it is necessary to go back to the conceptual definition of the db distribution expressed in terms of the Beta distribution (1). In terms of the conceptual definition, the mean and variance of a db distribution with shape parameters α and β may be written as

$$\mu = \frac{1}{\kappa}\sum_{i=0}^{n} i f((i+1)/(n+1))$$
$$\sigma^2 = \frac{1}{\kappa}\sum_{i=0}^{n} (i-\mu)^2 f((i+1)/(n+1)),$$

where $f(\cdot)$ is the probability density function of the Beta distribution with shape parameters α and β, n is equal to n_{top} (the maximum value of the support of the distribution), and κ is the normalizing constant

$$\kappa = \sum_{i=0}^{n} f((i+1)/(n+1)) \ .$$

In the foregoing, zero origin indexing (i.e., that ζ is TRUE) is assumed. This is of no real consequence given that the method is so rough to start with.

To get approximate expressions for the mean and variance of the distribution, one may manipulate the sums in the expressions for μ, σ^2, and κ into the form of Riemann sums that approximate integrals. This reveals that

$$\mu \approx \frac{(n+2)^2}{\kappa}\int_0^1 x f(x)\, dx - 1$$

and that

$$\kappa \approx (n+2)\int_0^1 f(x)\, dx \ .$$

The integral of $xf(x)$ is the mean of the associated Beta distribution, $\alpha/(\alpha+\beta)$ (Johnson et al. 1995, Chapter 25, equation 25.15a) and the integral of $f(x)$ is, of course, just 1, whence $\kappa \approx n+2$. Therefore,

$$\mu \approx \frac{(n+2)\alpha}{\alpha+\beta} - 1 \ .$$

Proceeding similarly, one finds that

$$\sigma^2 \approx \frac{(n+2)^2\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \ .$$

This latter result makes use of the fact that the variance of the associated Beta distribution is

$$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

(Johnson et al. 1995, Chapter 25, equation 25.15b).

To calculate the approximate method of moments estimates, which I shall denote by $\breve{\alpha}$ and $\breve{\beta}$ respectively, equate the foregoing approximate expressions for μ and σ^2 to the observed sample mean and variance ($\bar{x}$ and s^2) and solve for α and β. One obtains

$$\breve{\alpha} = \frac{(n+2)^2 a}{s^2(a+1)^3} - \frac{1}{a+1}$$
$$\breve{\beta} = a\breve{\alpha},$$

where for convenience, I have set $a = (n+1-\bar{x})/(1+\bar{x})$.

Note that although the Beta distribution is undefined for non-positive values of α and β, the foregoing approximate moment estimation procedure can be applied without problem to data for which non-positive parameter estimates are appropriate.

It must be emphasized here that the explicit moment estimation procedure discussed above is not intended to be applied by users. It is provided for the purpose of producing starting values for the maximum likelihood and "exact" moment estimation procedures. The estimates produced via the explicit moment estimation procedure are not very good, and in general, appear to be substantially biased. (See Figure 4.) Despite this, they seem to be adequate as starting values.

Quality of the estimators

I investigated the question of how well the estimation procedures perform by means of a number of simulation experiments.

Some confidence intervals for the parameters

In the first experiment, I determined interval estimates of α and β for a small grid of true values of these parameters. One hundred samples were generated for each combination of the true parameter values, and the means and standard errors of these estimates were calculated. Each sample was of size of 100. For each combination of the true parameters, 95% confidence intervals (mean $\pm 1.96\times$ standard error) for the individual parameters were then calculated. Table 1 displays the sample means of the estimates of α and β and the corresponding confidence intervals.

Half of the 18 confidence intervals failed to cover the true values. For the $(3,3)$ combination, the β interval failed to cover. For the $(3,6)$, $(6,6)$, $(9,3)$, and $(9,9)$ combinations, both intervals failed to cover. Where the confidence intervals failed to cover, they missed on the high side. As shall be seen later on, (Figure 6 in subsection **Asymptotic bias in the maximum likelihood estimates**), this would seem to be the expected behavior. The amounts by which the confidence intervals missed the true values were not egregiously large. The worst case was for the $(9,3)$ combination, where the lower endpoint of the confidence interval for α was greater than 9 by 0.294.

α	β	$\bar{\hat{\alpha}}$	95% CI for α	$\bar{\hat{\beta}}$	95% CI for β
3	3	3.014	(3.005, 3.023)	3.011	(3.003, 3.019)
3	6	3.029	(3.021, 3.038)	6.058	(6.041, 6.075)
3	9	3.018	(3.01, 3.026)	9.056	(9.031, 9.08)
6	3	6.056	(6.04, 6.072)	3.015	(3.007, 3.022)
6	6	6.029	(6.012, 6.046)	6.042	(6.025, 6.059)
6	9	6.019	(6.004, 6.035)	9.022	(8.999, 9.045)
9	3	9.002	(8.974, 9.029)	3.008	(2.998, 3.017)
9	6	9.032	(9.007, 9.057)	6.014	(5.997, 6.031)
9	9	9.032	(9.006, 9.057)	9.005	(8.979, 9.031)

Table 1: Some sample mean parameter estimates and 95% confidence intervals for the true values.

Comparison of estimators using mean squared error

I next conducted an experiment to investigate how well the two sorts of moment estimator compared with the maximum likelihood estimator. In this experiment, 100 samples, each of size 100, were generated (using `rdb()`) from distributions

$$\mathrm{db}(\alpha_i, \beta_j, n_{\mathrm{top}} = 10, \zeta = \mathtt{TRUE}),$$

with α_i and β_j varying over the set $\{0, 1, 2, ..., 9, 10\}$. For each of the three possible estimators (explicit but approximate method of moments, "exact" method of moments, and maximum likelihood), the mean squared error of the estimates, corresponding to the appropriate set of 100 samples, was calculated. The mean square error is defined as

$$\mathrm{MSE} = (\alpha - \bar{\alpha})^2 + (\beta - \bar{\beta})^2 + s_{\alpha}^2 + s_{\beta}^2 \; .$$

In the foregoing, $\bar{\alpha}$ and s_{α}^2 are the sample mean and variance of the 100 values of the estimates of α ($\breve{\alpha}_i$, $\tilde{\alpha}_i$, $\hat{\alpha}_i$ as the case may be) arising from the 100 samples generated for the given pair of parameter values. Similarly for $\bar{\beta}$ and s_{β}^2. A subset of the MSE values that were produced is presented in table 2.

It is worth mentioning that the MSEs of the "exact" moment estimates and of the maximum likelihood estimates were not adversely affected by the possibly poor starting values provided by the approximate moment estimates. In a simulation experiment, it is possible to use the *true* values of the parameters as starting values. Doing so gave rise to estimates that were virtually identical to those obtained when the approximate moment estimates were used to start the optimization.

A plot of the "exact" moment estimates against the maximum likelihood estimates is shown in Figure 3. The MSE for the "exact" moment estimates tracked the MSE for the maximum likelihood estimates almost exactly except for one striking outlier, corresponding to the parameter pair $\alpha = 10$, $\beta = 0$. A little bit of further experimentation indicated that this outlier is a one-off aberration, and that both "exact" moment estimation and maximum likelihood estimation are subject to occasional instability. More simulation experiments could be considered, but there are too many possibilities for this line of enquiry to be pursued in the current paper.

Unsurprisingly, the MSE of estimates produced by the approximate moment method did not track that of the maximum likelihood estimates nearly as closely, as shown in Figure 4. There is some

		Mean squared error		
α	β	Approx. Mom.	Exact Mom.	Max. Like.
0	0	1.24	0.10	0.10
5	0	55.58	8.13	8.03
10	0	636.59	102.20	45.55
0	2	4.93	0.69	0.71
5	2	1.10	1.10	1.09
10	2	9.24	7.11	6.83
0	4	27.60	2.50	2.43
5	4	1.02	1.02	1.06
10	4	3.19	3.28	3.45
0	6	94.64	8.58	8.53
5	6	1.34	1.34	1.37
10	6	3.08	3.06	3.13
0	8	292.97	28.21	28.08
5	8	1.48	1.47	1.45
10	8	3.53	3.53	3.69
0	10	556.54	50.80	52.78
5	10	3.82	3.80	3.93
10	10	5.40	5.40	5.78

Table 2: Subset of the MSE values plotted in Figures 3, 4 and 5.

suggestion that the MSE pairs are close to the "$y = x$" line for small values of MSE, but for values larger than about five, the plot of the MSE values becomes rather wild.

Evidence from the simulation experiments described here indicates that the MSE becomes large when the difference between α and β becomes large. Figure 5 displays a plot of the maximum likelihood MSE against $|\alpha - \beta|$. This figure indicates that the MSE values are all of roughly the same "moderate" size until the absolute difference becomes greater than or equal to six. At this point the values start to increase substantially.

Asymptotic bias in the maximum likelihood estimates

The squared bias component of the MSE for the maximum likelihood estimates was substantial. However, the simulation experiment described above used a sample size of 100 exclusively. To assess the impact of sample size on the bias in the estimates, I undertook a further simulation experiment in which I set α and β to have true values 0 and 10 respectively, which were the values that led, in the foregoing experiment, to the largest MSE. The sample size was allowed to range over the set $\{100, 200, 300, 500, 1000, 5000\}$. Five hundred simulated samples were generated in each instance, and the means and standard errors of the bias in parameter estimates were calculated.

Ninety-five percent confidence intervals for the mean bias at were plotted for both the α and β estimates. These plots are shown in Figure 6. This figure indicates that for this pair of true parameter values, the bias remains "significantly" different from 0 until the sample size reaches the surprisingly large value of 2000.

Examples

Example 1: Simulated binomial data

For an initial example, I simulated 100 data from a binomial distribution $\text{Bin}(10, 0.25)$ and fitted a db distribution to that. I set $\zeta =$TRUE and $n_{\text{top}} = 10$ (the "conceptual" upper bound). A plot of the resulting fit is shown in Figure 7. The fit is consistent with the other possible values of the probabilities of the various counts.

Example 2: The Downloads data

The Downloads data from Weiß (2018) consist of a time series (of length 267) of the observed daily number of downloads of a TEX editor for the period from June 2006 to February 2007. These data are

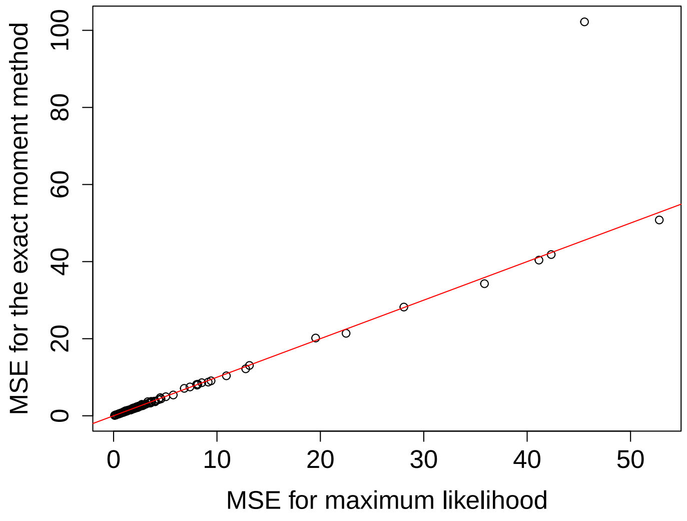


Figure 3: The MSE from "exact" moment estimation plotted against the MSE from maximum likelihood estimation. The superimposed red line is the line of "exact agreement" between the two estimators, i.e. it has slope 1 and intercept 0.

available as `Downloads` in the CRAN package **hmm.discnp**. They can also be obtained from the Wiley website `https://www.wiley.com/en-gb/An+Introduction+to+Discrete+Valued+Time+Series-p-9781119096962`, by clicking on "Downloads" and then on the "Download" button next to "Datasets". This will provide a zip archive of all of the data sets from Weiß's book.

Prima facie, it might seem plausible that these data are Poisson-distributed, but they are, in fact, much too overdispersed to be Poisson; the sample mean is 2.401, whereas the sample variance is 7.506. Weiß finds that an INAR(1) (integer-valued autoregressive, $p = 1$) model provides a good fit. In fitting a db distribution to these data, I took $\zeta =$ `TRUE` (zero counts are observed) and $ntop = 15$ (1+ the observed maximum of the data, which is 14). This db fit yields a mean of 2.451 and a variance of 7.461. A plot of this fit is shown in Figure 8. Of course, simply fitting a db distribution is not really appropriate since this treats the data as being i.i.d., and as Weiß's analysis shows, there is strong evidence of serial dependence in these data. Moreover, a goodness of fit test yields a p-value of 0.03 (see the help for `gof()` in the **dbd** package). One would thereby reject the hypothesis that the fit of the db distribution is adequate at the 0.05 significance level.

In other analyses of these data, the details of which it is inexpedient to discuss here, I have fitted hidden Markov models with marginal db distributions and varying numbers of states to these data. I used both AIC and BIC to select the number of states. Both criteria indicate that a two-state model is optimal, i.e., a model involving serial dependence is chosen over the model in which the data are i.i.d.

Example 3: The Sydney Coliform Count data

These data were analyzed in Turner et al. (1998). In that paper, the data were modeled after a certain transformation had been applied, as having a hidden Markov model structure with marginal Poisson distributions. I have since discretized these data into five (ordered) categories. The resulting data set is available as `SycColDisc` in the package **hmm.discnp**. I fitted db distributions to subsets of these data, treating them as having the numeric values $1, 2, \ldots, 5$, taking $n_{\text{top}} = 5$ and $\zeta =$ `FALSE`.

Plots of the fits, together with the observed proportions, are shown in Figure 9, for the Bondi East data at each of the four depths, 0, 20, 40, and 60 meters.

Example 4: The Sydney Coliform Count data (continued)

In general, it may be of interest to provide graphical representations of the uncertainty in the estimates of the parameters of db distributions. This can be done by plotting (say) 95% confidence ellipses around the point estimates. The **ellipse** package (Murdoch and Chow 2018) provides convenient means of plotting such ellipses.

In the context of the current example, it is also of interest to examine whether there are differences amongst the distributions associated with the various depth and location combinations. Such exami-

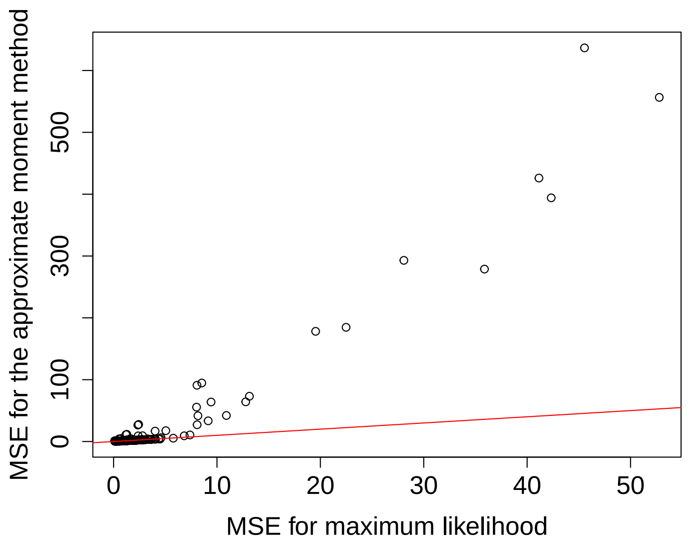


Figure 4: The MSE from approximate moment estimation plotted against the MSE from maximum likelihood estimation. The superimposed red line is the line of "exact agreement" between the two estimators, i.e. it has slope 1 and intercept 0.

nation can also be effected by means of plotting confidence ellipses. In this setting, one would plot confidence ellipses for the differences between the parameters corresponding to different combinations. Assuming that the samples are independent (possibly problematic here), the covariance matrix on which to base the confidence ellipse for the difference between the parameters is the sum of the two individual covariance matrices.

It is also possible to test the hypothesis that the distributions corresponding to a number of different combinations of depths and locations are all identical (again assuming the samples to be independent) by means of a likelihood ratio test. The foregoing ideas can be illustrated using the four different depths at the `"BondiE"` location. Confidence ellipses for the parameters corresponding to the four depths are shown in Figure 10. Confidence ellipses for the six pairwise differences are shown in Figure 11. Code for effecting the likelihood ratio test is as follows:

```
library(dbd)
X       <- hmm.discnp::SydColDisc
X$y     <- as.numeric(X$y)
X       <- split(X,f=with(X,interaction(locn,depth)))
X       <- X[c("BondiE.0","BondiE.20","BondiE.40","BondiE.60")]
fitz    <- lapply(X,function(x){mleDb(x$y,ntop=5)})
x.all   <- unlist(lapply(X,function(x){x$y}))
fit.all <- mleDb(x.all,ntop=5)
ll0     <- logLik(fit.all) # Two parameters.
ll1     <- sum(sapply(fitz,logLik)) # Eight parameters.
print(pchisq(2*(ll1-ll0),6,lower=FALSE)) # Df = 8 - 2.
```

The resulting p-value is 0.9781; i.e., there is no evidence at all of any differences. This conclusion is confirmed by Figure 11, wherein it can be seen that the 95% confidence ellipses all contain the point $(0,0)$. It is also in accordance with the visual impression given by Figure 9 in which the plots of the four distributions all look very similar.

An analogous exercise was done involving measurements all made at a depth of 60 meters, at four of the seven locations (Longreef, Bondi East, Malabar Offshore, and North Head Offshore; two "controls" and two "outfalls"). The likelihood ratio test yielded a p-value equal to 6.37×10^{-9}, i.e., effectively zero. The origin $(0,0)$ was exterior to the 95% confidence ellipses for the pairwise differences in four of the six instances.

Note that the foregoing analyses of the Sydney Coliform data are superficial in that they take no account of the serial dependence of these data. Undertaking analyses that accommodate serial dependence would lead us much too far afield.

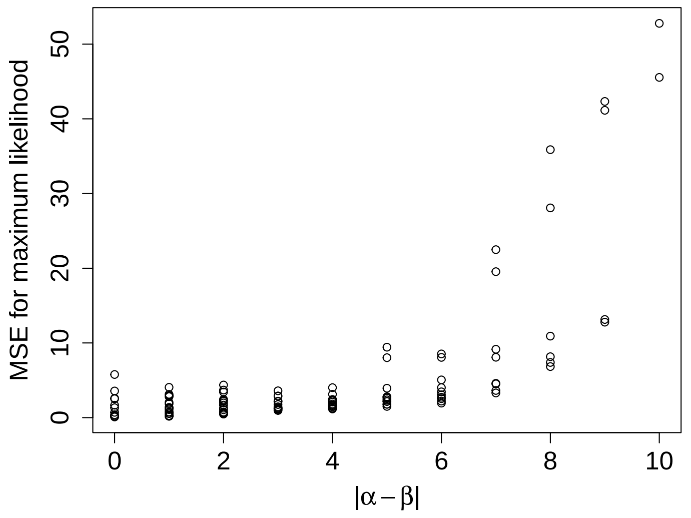


Figure 5: The MSE from maximum likelihood estimation plotted against the absolute value of the difference the parameters in the corresponding parameter pair.

Example 5: Monocyte counts and psychosis ratings

The monocyte counts and psychosis ratings data arose in a study initiated by Jonathan Williams, who was at the start of the study, working for the Northland District Health Board in New Zealand. The study is still ongoing, and the results have not yet been published. The data involved cannot be publicly released due to patient confidentiality issues.

This example is really what motivated the development of the db distribution. The data consist of pairs of sequences of observations of monocyte blood counts, discretized to a 1 to 5 scale, and ratings of severity of psychosis on a 0 to 4 scale. The observations were made on 1258 patients. They were made at irregularly spaced times, and there were varying numbers of observations per patient. The different types of sequence were not observed at the same times, and there were usually different numbers of observation between the types.

The analysis of these data was intricate, and the details cannot be gone into here. The crucial feature of the analysis is that hidden Markov models, whose marginal distributions were db distributions were fitted to both types of sequence. The value of n_{top} was taken to be 5 for the monocyte counts and 4 for the psychosis ratings, and that of ζ to be `FALSE` for the monocyte counts and `TRUE` for the psychosis ratings.

For each type, a three-state model was chosen (by means of a cross-validation technique). Plots of the db distributions corresponding to each of the three states are shown for the monocyte counts in Figure 12 and for the psychosis ratings in Figure 13.

Example 6: The Parsonnet scores from the cardiacsurgery data

As discussed in the introduction, I fitted both a db and a beta-binomial distribution to these data and conducted goodness of fit tests. In the case of the db distribution, I chose `ntop = 71` and set `zeta = TRUE` (since zeroes appear in the data). In the case of the beta-binomial distribution, I set `size = 71`. The value 71 is that which was used in the paper by Wittenberg (to appear in *Statistical Methods in Medical Research*) referred to on page 485.

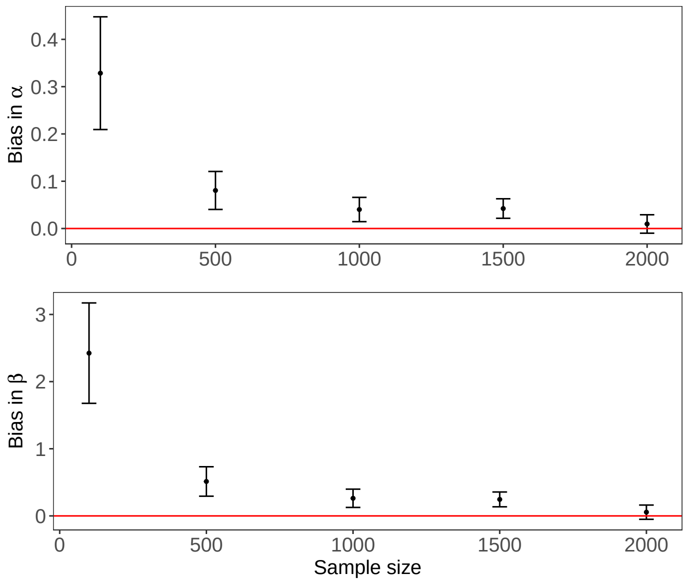


Figure 6: Ninety-five percent confidence intervals for the mean bias in estimates of $\alpha = 0$ and $\beta = 10$, plotted against the sample size. Five hundred sample were generated for each sample size. The red horizontal lines are the desired zero bias level.

```
# Note that the package spcadjust must be available.
data("cardiacsurgery", package = "spcadjust")
xxx  <- cardiacsurgery$Parsonnet
fit1 <- mleDb(xxx,ntop=71,zeta=TRUE)
g1   <- gof(fit1,obsd=xxx,MC=TRUE,verb=TRUE,seed=42)
fit2 <- mleBb(xxx,size=71)
g2   <- gof(fit2,obsd=xxx,MC=TRUE,verb=TRUE,seed=17)
```

Seeds were set in the calls to `gof()` in order for the Monte Carlo p-values to be reproducible. In both cases, the value of `nsim` was left at its default value of 99, which yielded a Monte Carlo p-value of 0.01. The values of the test statistic were huge, 12792.5 in the case of the db distribution, and 4004.668 in the case of the beta-binomial distribution. I therefore strongly suspect that if `nsim` had been increased to, say 9999, then the p-values would have been 0.0001. Howeveri, I have not checked this out.

Plots of the fits (not shown) can be produced by:

```
plot(fit1,obsd=xxx,main="db")
plot(fit2,obsd=xxx,main="beta-binomial)
```

These plots reveal that there are indeed substantial discrepancies between the fitted probabilities and the observed proportions. The fitted probabilities from the dbd distribution, although differing significantly from the observed proportions, are "not too different" from each other. This can be seen from the plot (again not shown) that can be produced by:

```
x1   <- plot(fit1,plot=FALSE)
x2   <- plot(fit2,plot=FALSE)
ylim <- range(x1$p,x2$p)
plot(fit1,main="Comparing db and beta-binomial fits",ylim=ylim)
with(x2, lines(x+0.5,p,type="h",col="blue"))
legend("topright",lty=1,col=c("red","blue"),
       legend=c("db","beta-binomial"),bty="n")
```

The large size (5595), of the data set that is involved here, revealed an interesting timing issue. The goodness of fit test took a great deal (of the order of 50 times) longer for the beta-binomial distribution than for the db distribution. Some rudimentary timing experiments revealed that for data sets of this size, the random number generator `rbetabinom()` from the **rmutil** package is about 50 times slower than `rdb()` from the **dbd** package. I have not investigated the reason for this phenomenon.

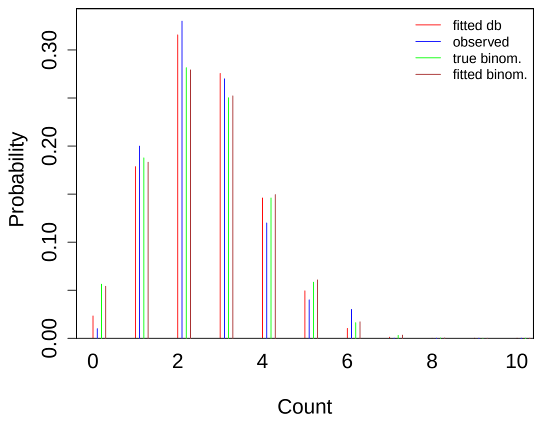


Figure 7: Fit of a db distribution to a sample from a Bin(10, 0.25) distribution. Also shown are the observed proportions, the true binomial probabilities, and the fitted binomial probabilities.

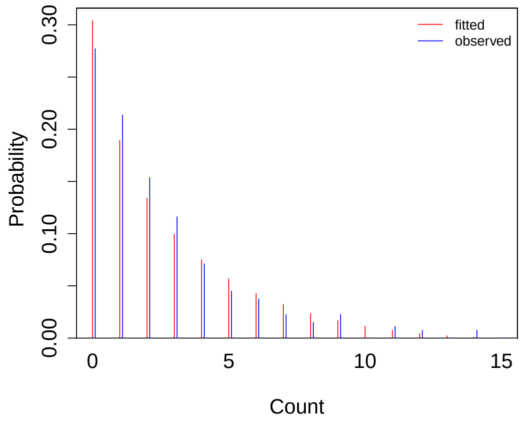


Figure 8: Fit of a db distribution to the Downloads data from Weiß (2018). Also shown are the observed proportions.

Example 7: Underdispersed data

As mentioned in the introduction, the **dbd** package provides two data sets which are of the nature of binomial data but appear to be underdispersed relative to the binomial distribution. Here are some examples which illustrate fitting db and beta-binomial distributions to these data.

The horse race prediction data from the **dbd** package (see the help for `hrsRcePred`) provides four examples.

```
library(dbd)
X     <- hrsRcePred
top1e <- X[X$sbjType=="Expert","top1"]
top1n <- X[X$sbjType=="NonXpert","top1"]
top3e <- X[X$sbjType=="Expert","top3"]
top3n <- X[X$sbjType=="NonXpert","top3"]
fit1e <- mleDb(top1e,ntop=10,zeta=TRUE)
fit1n <- mleDb(top1n,ntop=10,zeta=TRUE)
fit3e <- mleDb(top3e,ntop=10,zeta=TRUE)
fit3n <- mleDb(top3n,ntop=10,zeta=TRUE)
```

The ratios of the raw variance to the putative binomial distribution variance are 0.4895, 0.4100, 0.5718

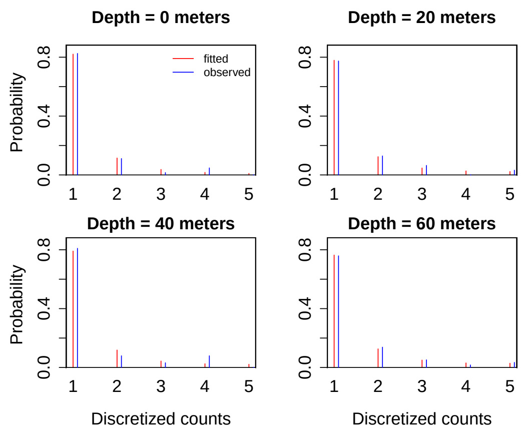


Figure 9: Fits of db distributions to discretized Sydney coliform count data from the Bondi East location, at depths equal to 0, 20, 40, and 60 meters. Also shown are the observed proportions.

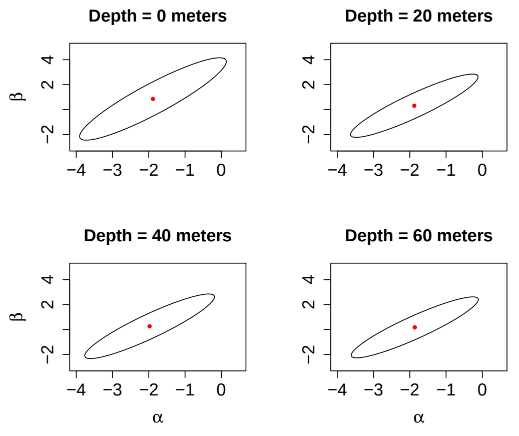


Figure 10: Ninety-five percent confidence ellipses for the parameters of the db distributions fitted to the data from the Bondi East location at depths 0, 20, 40, and 60 meters.

and 0.7745 respectively. All ar substantialy less than 1, whence these data sets appear to be indeed underdispersed.

The fitting procedures proceeded without complaint for all four data sets. Goodness of fit tests (two of which required increasing `maxit` from its default value) indicate adequate fit for three of the four data sets.

```
# Set seeds to get repeatable Monte Carlo p-values.
pv1e <- gof(fit1e,obsd=top1e,MC=TRUE,maxit=5000,
            seed=49,verb=TRUE)$pval               # 0.02
pv1n <- gof(fit1n,obsd=top1n,MC=TRUE,
            seed=128,verb=TRUE)$pval              # 0.79
pv3e <- gof(fit3e,obsd=top3e,MC=TRUE,
            seed=303,verb=TRUE)$pval              # 0.35
pv3n <- gof(fit3n,obsd=top3n,MC=TRUE,maxit=3000,
            seed=24,verb=TRUE)$pval               # 0.40
```

There is significant evidence that the db distribution is not appropriate for the `top1e` data (p-value $= 0.02$). For the other three data sets, the p-values are all large, indicating that there is no evidence of any problems with any of these fits.

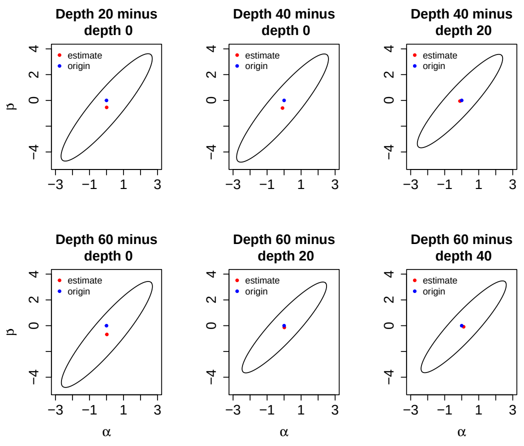


Figure 11: Ninety-five percent confidence ellipses for pairwise differences between the parameter vectors of the db distributions fitted to the data from the Bondi East location at depths 0, 20, 40, and 60 meters.

Note that for the problematic fit (i.e., `fit1e`), the parameter estimates might be considered to be excessively large:

```
    alpha      beta
145.14659  21.23249
```

This may be indicative of problems.

The beta-binomial distribution fits acceptably to all four of the data sets `top1e`, `top1n`, `top3e`, and `top3n`. For example, for the first of these data sets (for which the fit of the db distribution was rejected), the following code produces a Monte Carlo p-value of 0.11:

```
bbfit1e <- mleBb(top1e,size=10)
bbpv1e  <- gof(bbfit1e,obsd=top1e,MC=TRUE,maxit=5000,
                seed=792,verb=TRUE)$pval                # 0.11
```

(For the other three data sets, Monte Carlo p-values of 0.64, 0.62, and 0.75 were obtained.)

Another two examples of underdispersed data are provided by the visual recognition data from the **dbd** package (see the help for `visRecog`). The ratios of the raw variance to the putative binomial distribution variance for these data are 0.7635 and 0.7979. The Monte Carlo p-values from fitting the db distribution were 0.92 and 0.71, and those from fitting the beta-binomial distribution were 0.97 and 0.83.

As a "reality check", it is worth noting that fitting a simple binomial distribution to these data sets yielded Monte Carlo p values, from goodness of fit tests, equal to 0.13, 0.61, 0.72, and 0.70 for the `hrsRcePred` data, and 0.98 and 0.81 for the `visRecog` data. That is, binomial distributions fit these data sets acceptably despite their apparent underdispersion.

Concluding remarks

The db distribution is a new distribution that can be applied to any sort of data that takes values in a finite discrete set. It is an ad hoc distribution and does not require any theoretical justification in terms of properties that the data may have. It is very flexible, with the restriction that (as remarked in the **Introduction**) it is effectively unimodal. The values of the distribution are integers varying from 0 to n or from 1 to n for some n, and data to which a db distribution is to be fitted must be converted (recoded) into that form. In order that the fit should make practical sense, the data should, generally speaking, bear some relation to counts or at least be ordered. However, there is no theoretical requirement that this should be the case.

A number of examples have been given in this paper illustrating the fit of the db distribution to different data sets. These examples show that the db distribution may reasonably be expected to be useful to data analysts who need to deal with discrete data that do not conform to one of the standard distributions.

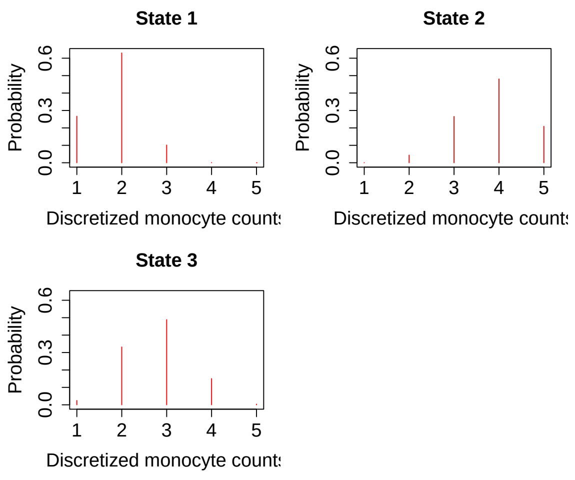


Figure 12: Marginal db distributions corresponding to each of the three states of a hidden Markov model fitted to the monocyte count data.

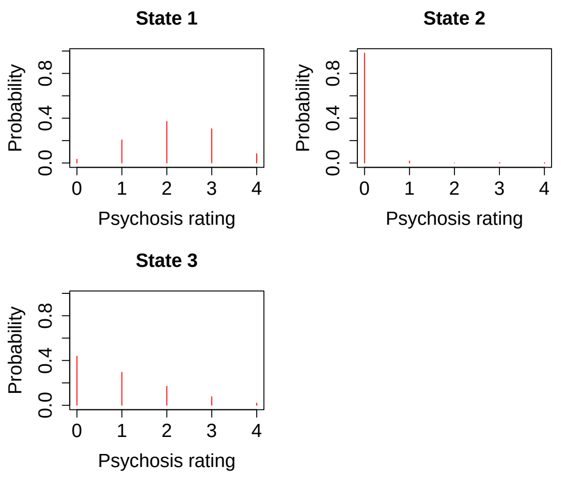


Figure 13: Marginal db distributions corresponding to each of the three states of a hidden Markov model fitted to the psychosis rating data.

Appendix I

Here I derive, from the conceptual definition (1) of the db distribution, an expression for the PMF of this distribution which, for fixed values of the support parameters n_{top} and ζ, is in exponential family form. A distribution is in the exponential family if its probability density or mass function has a particular structure. Different authors and books express this structure in a variety of equivalent ways. (See e.g., Cox and Hinkley 1974, p.94, Davidson 2003, p. 168, Hogg et al. 2005, p. 400. Liero and Zwanzig 2012, p. 15, Abramovich and Ritov 2013, p. 13. The reader may also find it useful to access `https://en.wikipedia.org/wiki/Exponential_family`.) Almost all of the commonly used distributions (with the notable exception of the uniform distribution) are in the exponential family.

A suitable expression for the exponential family form of a probability density or mass function, of a (scalar) distribution depending on a parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)^\top$, is

$$f(x \mid \boldsymbol{\theta}) = h(x) \exp \left(\sum_{i=1}^{k} \eta_i(\boldsymbol{\theta}) T_i(x) - A(\boldsymbol{\theta}) \right) .$$

The "natural parameters" of the distribution are the $\eta_i(\boldsymbol{\theta})$.

The pdf of the Beta distribution can be written in exponential family form (with natural parameters equal to α and β) as

$$f(x \mid \alpha, \beta) = h_{\mathrm{B}}(x) \exp\{\alpha \log(x) + \beta \log(1-x) - \log B(\alpha, \beta)\},$$

where $h_{\mathrm{B}}(x) = (x(1-x))^{-1}$, and $B(\alpha, \beta)$ is the beta function which is equal to $\Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$ (where in turn $\Gamma(\cdot)$ is the gamma function).

The PMF of the db distribution, expressed in terms of the Beta distribution (1), is

$$\Pr(X = x \mid \alpha, \beta) = \frac{1}{\kappa} f\left(\frac{x - n_{\mathrm{bot}} + 1}{n_{\mathrm{top}} - n_{\mathrm{bot}} + 2}\right),$$

where $f(\cdot)$ is the pdf of the Beta distribution and

$$\kappa = \sum_{i=n_{\mathrm{bot}}}^{n_{\mathrm{top}}} f\left(\frac{i - n_{\mathrm{bot}} + 1}{n_{\mathrm{top}} - n_{\mathrm{bot}} + 2}\right) .$$

Hence, if one sets $h(x) = h_{\mathrm{B}}((x - n_{\mathrm{bot}} + 1)/(n_{\mathrm{top}} - n_{\mathrm{bot}} + 2))$, $T_1(x) = \log((x - n_{\mathrm{bot}} + 1)/(n_{\mathrm{top}} - n_{\mathrm{bot}} + 2))$, and $T_2(x) = \log((n_{\mathrm{top}} - x + 1)/(n_{\mathrm{top}} - n_{\mathrm{bot}} + 2)$, then the PMF of the db distribution can be written as

$$\Pr(X = x \mid \alpha, \beta) = h(x) \exp\left\{\alpha T_1(x) + \beta T_2(x) - \log\left[\sum_{i=n_{\mathrm{bot}}}^{n_{\mathrm{top}}} h(i) \exp\{\alpha T_1(i) + \beta T_2(i)\}\right]\right\}$$

Note that the $\log B(\alpha, \beta)$ terms, present in $f(\cdot)$, cancel when

$$f\left(\frac{x - n_{\mathrm{bot}} + 1}{n_{\mathrm{top}} - n_{\mathrm{bot}} + 2}\right)$$

is divided by κ.

The foregoing expression for the PMF is equal to

$$\Pr(X = x \mid \alpha, \beta) = h(x) \exp\{\alpha T_1(x) + \beta T_2(x) - A(\alpha, \beta)\},$$

where

$$A(\alpha, \beta) = \log\left(\sum_{i=n_{\mathrm{bot}}}^{n_{\mathrm{top}}} h(i) \exp\{\alpha T_1(i) + \beta T_2(i)\}\right) .$$

This is the expression given in (2) and is of exponential family form, although the constant $A(\alpha, \beta)$ might appear to be somewhat unorthodox.

Obviously, the value of $A(\alpha, \beta)$ is such that the values of $\Pr(X = i \mid \alpha, \beta)$, $i = n_{\mathrm{bot}}, \ldots, n_{\mathrm{top}}$, sum to 1.

Appendix II

When the PMF of a db distribution is expressed in the form (2), it is a relatively simple matter to derive an analytic expression for the gradient of the log-likelihood. Such an expression can be passed to `optim()` obviating the need for approximating the gradient numerically via finite differencing. The derivation of an analytic expression for the Hessian is equally easy. The `optim()` function makes no provision for using an analytically calculated Hessian. However, the availability of such an expression permits the calculation or estimation of the covariance matrix of the parameter estimates in an analytic manner. The derivations of the expressions for the gradient and Hessian are as follows.

The log-likelihood is

$$\begin{aligned}\ell &= \log \Pr(X = x \mid \alpha, \beta, n_{\mathrm{top}}, \zeta) \\ &= \log h(x) + \alpha T_1(x) + \beta T_2(x) - A(\alpha, \beta) .\end{aligned}$$

Consequently, the gradient is given by

$$\frac{\partial \ell}{\partial \alpha} = T_1(x) - \frac{\partial A}{\partial \alpha} \quad \frac{\partial \ell}{\partial \beta} = T_2(x) - \frac{\partial A}{\partial \beta} .$$

The Hessian is given by

$$\frac{\partial^2 \ell}{\partial \alpha^2} = -\frac{\partial^2 A}{\partial \alpha^2} \quad \frac{\partial^2 \ell}{\partial \alpha \partial \beta} = -\frac{\partial^2 A}{\partial \alpha \partial \beta}$$
$$\frac{\partial^2 \ell}{\partial \beta^2} = -\frac{\partial^2 A}{\partial \beta^2}$$

Now let

$$E = \exp(A) = \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) \exp\{\alpha T_1(i) + \beta T_2(i)\} \,.$$

Clearly

$$\frac{\partial A}{\partial \alpha} = \frac{1}{E}\frac{\partial E}{\partial \alpha} \quad \frac{\partial A}{\partial \beta} = \frac{1}{E}\frac{\partial E}{\partial \beta}$$
$$\frac{\partial^2 A}{\partial \alpha^2} = \frac{1}{E}\frac{\partial^2 E}{\partial \alpha^2} - \frac{1}{E^2}\left(\frac{\partial E}{\partial \alpha}\right)^2 \quad \frac{\partial^2 A}{\partial \alpha \partial \beta} = \frac{1}{E}\frac{\partial^2 E}{\partial \alpha \partial \beta} - \frac{1}{E^2}\left(\frac{\partial E}{\partial \alpha}\frac{\partial E}{\partial \beta}\right)$$
$$\frac{\partial^2 A}{\partial \beta^2} = \frac{1}{E}\frac{\partial^2 E}{\partial \beta^2} - \frac{1}{E^2}\left(\frac{\partial E}{\partial \beta}\right)^2 \,.$$

It remains to calculate the relevant partial derivatives of E. These are given by:

$$\begin{aligned}
\frac{\partial E}{\partial \alpha} &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) T_1(i) \exp(\alpha T_1(i) + \beta T_2(i)) \\
\frac{\partial E}{\partial \beta} &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) T_2(i) \exp(\alpha T_1(i) + \beta T_2(i)) \\
\frac{\partial^2 E}{\partial \alpha^2} &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) T_1(i)^2 \exp(\alpha T_1(i) + \beta T_2(i)) \\
\frac{\partial^2 E}{\partial \alpha \partial \beta} &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) T_1(i) T_2(i) \exp(\alpha T_1(i) + \beta T_2(i)) \\
\frac{\partial^2 E}{\partial \beta^2} &= \sum_{i=n_{\text{bot}}}^{n_{\text{top}}} h(i) T_2(i)^2 \exp(\alpha T_1(i) + \beta T_2(i)) \,.
\end{aligned}$$

The foregoing calculations have been translated into R code in the (undocumented) functions `gradDb()` and `hessDb()` in the **dbd** package.

Appendix III

Covariance matrices of the maximum likelihood estimates of the parameters of a db distribution may be calculated both by theoretical means (using, e.g., `aHess()` from the **dbd** package) or by a Monte Carlo procedure using the `mcCovMat()` function from the same package. Doing such calculations in a number of examples has indicated that there can be noticeable discrepancies between the theoretical and Monte Carlo results. To investigate this issue further, I calculated the variance of $\hat{\beta}$ from *known* (rather than estimated) parameters using both the theoretical (inversion of the Fisher information matrix) and Monte Carlo procedures. The essential part of the code used to do this is as follows.

```
obj        <- makeDbdpars(alpha=3,beta=3,ntop=10,zeta=TRUE,ndata=<some value>)
varBeta.mc <- mcCovMat(obj,nsim=500)[2,2]
varBeta.fi <- solve(do.call(finfo,obj))[2,2]
```

I effected the calculations for a range of sample sizes ("`ndata`"). The results are plotted in Figure 14.

The behavior depicted in Figure 14 is typical. The theoretical covariance matrices for the parameter estimates generally include variance entries which (for relatively small sample sizes) are appreciably smaller than the corresponding entries of the covariance matrices produced by Monte Carlo methods. Since the Monte Carlo covariance matrices are unbiased estimates of the true covariances, it would appear that the theoretical variances tend to underestimate the truth. This phenomenon is not peculiar to the db distribution. Such underestimation occurs in the context of the Beta distribution and very likely in other contexts as well. As illustrated by Figure 14, the level of underestimation (as would be expected) diminishes as the sample size increases. In the illustrated instance, the underestimation effectively disappeared when the sample size reached 200.

The fact that variances are underestimated by the theoretical covariance estimates implies that inference about the shape parameters based on the theoretical values should be treated with a certain amount of circumspection. Unless the sample size is large, confidence intervals may be somewhat too

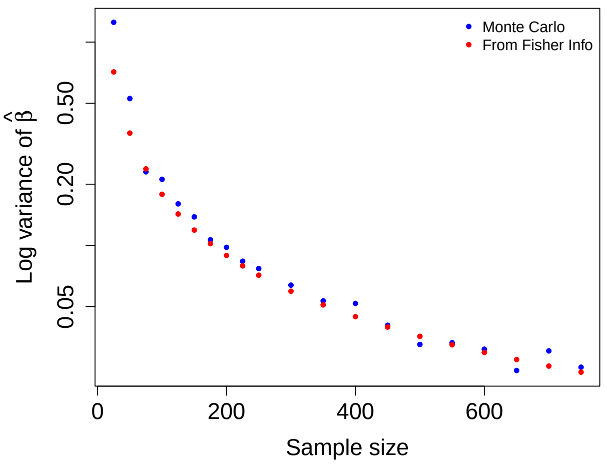


Figure 14: Monte Carlo and analytic estimates of the log variance of $\hat{\beta}$ for various sample sizes.

narrow, and hypothesis tests too liberal. When conducting inference about the shape parameters, it is advisable to estimate the covariance matrix using `mcCovMat()`. The procedure is not too computationally demanding and is thus reasonably quick in normal circumstances. It is probably a good idea, in circumstances in which inferential conclusions are critical, to calculate a number of Monte Carlo covariance matrix estimates (using different seeds) and to compare these with each other and with the "analytic" value of the covariance matrix.

The difference between results from using an "analytic" covariance matrix and those from using a Monte Carlo covariance matrix is also illustrated in Figure 15. The examples used are from the Bondi East Sydney Coliform Count data. (See Figures 10 and 11.) The chosen examples evince the most striking difference between the confidence ellipses based on the two different calculation methods.

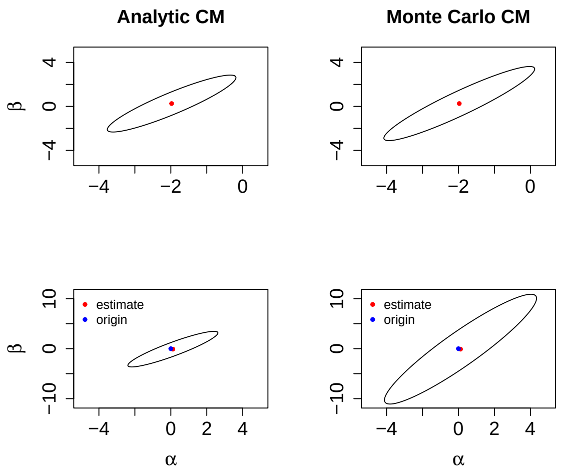


Figure 15: The top two panels show 95% confidence ellipses for the parameters of the db distribution for the Bondi East data at depth equal to 40 meters. The left-hand panel is based on the "analytic" covariance matrix, the right-hand panel on a Monte Carlo covariance matrix. The bottom two panels show 95% confidence ellipses for the difference in parameters between depth 60 meters and depth 40 meters. Again, the left-hand panel is based on the "analytic" covariance matrix and the right-hand panel on a Monte Carlo covariance matrix. The inferential conclusions with respect to the differences are unchanged.

Acknowledgements

The author would like to thank an anonymous referee for useful comments which led to substantial improvements in the paper.

The author also wishes to acknowledge the use of the New Zealand eScience Infrastructure (NeSI) high-performance computing facilities, which made possible the analysis of the monocyte counts and psychosis ratings discussed in **Example 5**. New Zealand's national facilities are provided by NeSI and funded jointly by NeSI's collaborator institutions and through the Ministry of Business, Innovation & Employment's Research Infrastructure programme. URL https://www.nesi.org.nz.

Bibliography

F. Abramovich and Y. Ritov. *Statistical Theory: A Concise Introduction*. CRC Press, Boca Raton, 2013. [p501]

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover, New York, 1972. [p485]

A. Baddeley, E. Rubak, and R. Turner. *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press, London, 2015. [p490]

D. R. Cox and D. V. Hinkley. *Theoretical Statistics*. Chapman and Hall, London, 1974. [p501]

A. C. Davidson. *Statistical Models*. Cambridge University Press, Cambridge, 2003. [p501]

A. Gandy and J. T. Kvaloy. Guaranteed conditional performance of control charts via bootstrap methods. *Scandinavian Journal of Statistics*, 40:647–668, 2013. doi: 10.1002/sjos.12006. [p486]

R. V. Hogg, J. W. McKean, and A. T. Craig. *Introduction to Mathematical Statistics*. Pearson/Prentice Hall, Upper Saddle River, New Jersey, 2005. [p501]

N. L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. Wiley, New York, 1995. [p485, 491]

H. Liero and S. Zwanzig. *Introduction to the Theory of Statistical Inference*. CRC Press, Boca Raton, 2012. [p501]

D. Murdoch and E. D. Chow. *ellipse: Functions for Drawing Ellipses and Ellipse-Like Confidence Regions*, 2018. URL https://CRAN.R-project.org/package=ellipse. R package version 0.4.1. [p494]

V. Parsonnet, D. Dean, and A. D. Bernstein. A method of uniform stratification of risk for evaluating the results of surgery in acquired adult heart disease. *Circulation*, 79:I3 – I12, 1989. [p486]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL https://www.R-project.org/. [p485]

J. G. Skellam. A probability distribution derived from the binomial distribution by regarding the probability of success as variable between the sets of trials. *Journal of the Royal Statistical Society, Series B*, 10:257 – 261, 1948. [p486]

S. H. Steiner, R. J. Cook, V. T. Farewell, and T. Treasure. Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, 1:441 – 452, 2000. [p486]

B. Swihart and J. Lindsey. *rmutil: Utilities for Nonlinear Regression and Repeated Measurements Models*, 2020. URL https://CRAN.R-project.org/package=rmutil. R package version 1.1.4. [p486]

R. Turner. *hmm.discnp: Hidden Markov models with discrete non-parametric observation distributions*, 2020. R package version 3.0-6. [p485]

R. Turner. *The db Distribution*, 2021. R package version 0.0-22. [p489]

T. R. Turner, M. A. Cameron, and P. J. Thomson. Hidden Markov chains in generalized linear models. *Canadian Journal of Statistics*, 26:107 – 125, 1998. [p494]

C. H. Weiß. *An Introduction to Discrete-Valued Time Series*. John Wiley & Sons, Chichester, 2018. [p493, 498]

Rolf Turner
Honorary Research Fellow
Department of Statistics
The University of Auckland
New Zealand
ORCID: 0000-0001-5521-5218
`r.turner@auckland.ac.nz`

Automatic Time Series Forecasting with Ata Method in R: ATAforecasting Package

by Ali Sabri Taylan, Güçkan Yapar and Hanife Taylan Selamlar

Abstract Ata method is a new univariate time series forecasting method that provides innovative solutions to issues faced during the initialization and optimization stages of existing methods. The Ata method's forecasting performance is superior to existing methods in terms of easy implementation and accurate forecasting. It can be applied to non-seasonal or deseasonalized time series, where the deseasonalization can be performed via any preferred decomposition method. The R package **ATAforecasting** was developed as a comprehensive toolkit for automatic time series forecasting. It focuses on modeling all types of time series components with any preferred Ata methods and handling seasonality patterns by utilizing some popular decomposition techniques. The **ATAforecasting** package allows researchers to model seasonality with STL, STLplus, TBATS, stR, and TRAMO/SEATS, and power family transformation and analyze the any time series with a simple Ata method and additive, multiplicative, damped trend the Ata methods and level fixed Ata trended methods. It offers functions for researchers and data analysts to model any type of time series data sets without requiring specialization. However, an expert user may use the functions that can model all possible time series behaviors. The package also incorporates types of model specifications and their graphs, uses different accuracy measures that surely increase the Ata method's performance.

Introduction

Ata method (Cetin and Yavuz, 2020; Yilmaz et al., 2019; Yapar et al., 2019; Yapar, 2018; Yapar et al., 2018, 2017) is a new univariate time series forecasting method which provides innovative solutions to issues faced during the initialization and optimization stages of existing methods. **ATAforecasting** performance is superior to existing methods both in terms of easy implementation and accurate forecasting. It can be applied to non-seasonal or deseasonalized time series, where the deseasonalization can be performed via any preferred decomposition method. This methodology performed extremely well on the M3 and M4-Competition data.

The original exponential smoothing has accomplished well in a wide range of practical researches, and it is well built as a precise and optimal forecasting method. Nonetheless, two essential difficulties are to choose the smoothing constant and starting value in exponential smoothing theory. The Ata method suggests an alternative method for smoothing constant and initial value. The Ata method places more emphasis than the classical method on most recent activities. The forecasting error is compared to the error in forecasts obtained by the original model.

Exponential smoothing (ES) is not the only model. In fact, a family of models. ES models suppose that a time series has four components: seasonality, trend, level, and remainder. Bergmeir et al. (2016) recommended the bootstrap aggregation of ES methods. The bootstrap aggregation employs a Box–Cox transformation afterwards an a seasonal trend decomposition based on LOESS (LOcally Estimated Scatter-plot Smoother) (STL) to segregate the time series sub three part: remainder, seasonal and trend. The remainder is then bootstrapped via a moving block, and a new data is gathered via this bootstrapped residual part. Thereafter, an ensemble of ES models is calculated with the bootstrapped series.

Incorporating other types of model specifications and using different accuracy measures will surely increase the Ata method's performance. Like other approaches, the method can also benefit from certain transformations and decompositions of other types of more involved combinations, outlier detection, and other more complicated model selection strategies. The fact that these simple selection and combination strategies can perform better than existing methods is fascinating, and this further strengthens the idea that simplicity is indeed a prerequisite for forecasting accuracy.

Function	Description
ATA	Time series analysis and forecasting using the Ata method.
ATA.Core	The core algorithm of the Ata method.
ATA.Forecast	Produces forecasts from the output of ATA function.
ATA.Accuracy	Computes fitting and forecasting accuracy measures.
ATA.Transform	Computes transformed data using power transformation techniques.
ATA.BackTransform	Computes back transformed data using power transformation techniques.
ATA.BoxCoxAttr	Assigns attributes set for unit root and seasonality tests.
ATA.Seasonality	Tests seasonality.
ATA.Decomposition	Decomposes a time series into seasonal, trend, and irregular.
ATA.SeasAttr	Assigns attributes set for unit root and seasonality tests.
ATA.Plot	Specialized plot function of the output of ATA function.
ATA.Print	Specialized print screen function of the output of ATA function.
ATA.CI	Confidence interval function for the Ata method forecasts.

Table 1: A summary of the functions available in the **ATAforecasting** package.

Several decomposition techniques are used before applying the Ata method for selecting an optimized model. The high performance of these combined models is indicated with an empirical practice. The Ata method is analyzed in contrast practically with the most well-known forecasting techniques based on ES and ARIMA in accordance with its predictive performance on the M3 (Makridakis and Hibon, 2000) and M4-Competitions (Makridakis et al., 2018) data set and is illustrated to outperform its contestants.

Over the past few years, the preliminary research on ES (Brown, 1959; Pegels, 1969; Gardner and McKenzie, 1985) expanded to an approach based on a model so that there are 30 potential ES models for various types of trend, seasonality, and errors. The well-known of these are the simple ES, Holt's linear trend model, and Holt-Winter's model. Then, Gardner and McKenzie (1985) proposed damped trend model to help deal with overtrending. The reputation and universality of ES can also be attributed to its proven record against more sophisticated techniques (Makridakis et al., 1984; Makridakis and Hibon, 2000; Koning et al., 2005). The **forecast** package (Hyndman et al., 2020) in the programming language R (R Core Team, 2016) means that a fully automated software for fitting ETS models is available. These have led to a broadly appropriate ES modelling background, and with the use of latterly developed software packages, these ES models handle seasonality, trend, and other attributes of series without any human intervention (Hyndman et al., 2002, 2008; Hyndman and Athanasopoulos, 2019).

The Theta method (Assimakopoulos and Nikolopoulos, 2000) was introduced as a new univariate forecasting method which is similar to a simple ES model with drift, and its performance in terms of forecasting accuracy was prominent in M3-Competition. As confirmed once again in Assimakopoulos and Nikolopoulos (2000), it is well known that combining forecasts (Bates and Granger, 1969; Clemen, 1989) under certain circumstances improves forecasting accuracy (Armstrong, 1989, 2001; Makridakis and Winkler, 1983; Makridakis et al., 1982). Due to this, the research focuses on transformations, decompositions, rules, and combinations of ES and ARIMA (a few examples are (Clemen, 1989; Cleveland et al., 1990; Adya et al., 2000) to improve the forecasting performance rather than suggesting new forecasting methods.

Several other studies that are based on automatic forecasting procedures exist. Particularly for seasonal time series, the **forecast** package offers the TBATS model (Livera et al., 2011). TBATS uses a parsimonious trigonometric representation of seasonality instead of conventional seasonal indices and also incorporates ARMA errors. In addition, the function also automatically performs Box-Cox transformation of the time series if necessary.

This study introduces **ATAforecasting** (available from the Comprehensive R Archive Net- work at `https://cran.r-project.org/package=ATAforecasting`), a software application for R which performs a novel decomposition and power transformation-based approaches to time series forecasting using Ata method without any academic expertise. To sum up, the **ATAforecasting** package (Taylan et al., 2021a) provides a novel R interface for researchers interested in automatic time series analysis and students and academics who teach courses related to univariate time series analysis topics. There are main 13 functions available in the **ATAforecasting** package; see Table 1. We are going to describe all of them as we go on to explain the theoretical procedure. The rest of the paper is organized as follows. Section 2 presents a novel forecasting approach using the Ata method, gives an overview of the main estimation methods of the Ata method, and provides some technical details about the **ATAforecasting** package. Section 3 illustrates M-forecasting Competition dataset examples showing the package's functionality. Section 4 contains some concluding remarks.

Methodology

The objective of this study is to introduce a new decomposition-based approach to time series forecasting with the Ata method to provide the automation and the optimization of the Ata method which is an innovative and accurate univariate time series analysis method without any expertise of the R program. Specifically, we propose an analytical methodology of time series method with **ATAforecasting** R package, as it combines several stationarities and seasonality tests, Box–Cox transformations, seasonal decomposition techniques with the Ata method. We merge the various preceding concepts to attain a robust and broadly practicable automatic forecasting algorithm. The methodology involves 2 alternative algorithms with 6 steps as described and summarized below:

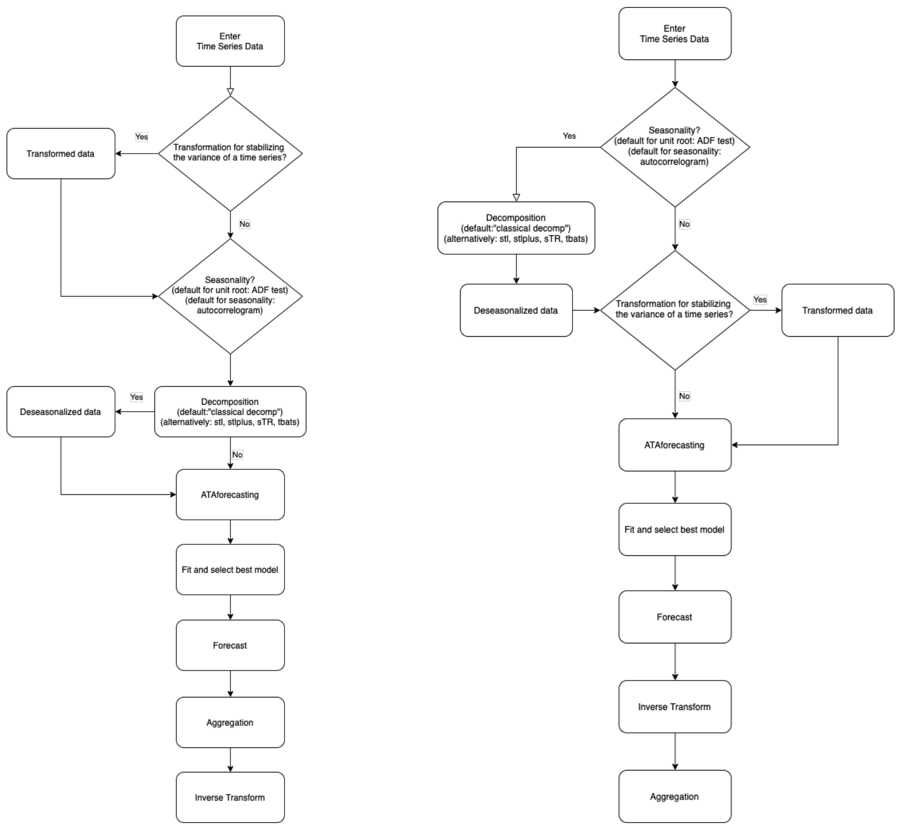
***First Algorithm* in Figure 1a**

- Step 1 Transformation
  Transformation for stabilizing the variance of a time series if necessary;
  There are many power transformation methods available to stabilizing linearity and variance. In this paper, logarithm, logarithm with shift parameter, Box–Cox, Box–Cox with shift parameter, Modulus, Bickel–Doksum, Dual, Yeo–Johnson, generalized logarithm (glog), and glog with power function (gpower) methods are able to applied.
- Step 2 Seasonality Test
  Identify and correct for seasonality in time series;
  There are several methods to detect stationarity and seasonality in time series. In this package, Augmented Dickey–Fuller (ADF), Phillips and Perron (PP), Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) unit root tests are adopted for stationarity. Autocorrelation function (ACF), Canova–Hansen (CH), Hylleberg-Engle-Granger-Yoo (HEGY), Osborn, Chui, Smith, and Birchenhall (OCSB) seasonal unit root tests are adopted for seasonality.
- Step 3 Decomposition
  Decompose the series into three components: trend, seasonal, and remainder;
  There are a few techniques to decompose time series. In this package, classical decomposition (decompose – **stats** package (R Core Team, 2016)), the Seasonal-Trend decomposition using LOESS (STL – **stats** package (R Core Team, 2016)), an enhanced of STL method (**stlplus** package (Hafen, 2016)), Trigonometric Seasonal Box–Cox Transformation–ARMA residuals–Trend and Seasonality (TBATS – **forecast** package (Hyndman et al., 2020)), Seasonal–Trend Decomposition Procedure Based on Regression (**stR** package (Dokumentov and Hyndman, 2018)) are adopted.
- Step 4 ATA Forecasting
  Apply ATA forecasting method to generate forecasts for the time series;
  Although there are many forecasting techniques available to perform (e.g., ETS, ARIMA, Theta, etc.), the Ata forecasting method is used. Ata method is an innovative new forecasting technique where the forms of the models are similar to exponential smoothing models. Still, the smoothing parameters depending on the sample size are optimized in a discrete space. Initialization is easier as it is done simultaneously when the parameters are optimized and less influential since the weights assigned to initial values approach zero quickly.
- Step 5 Selection and Aggregation
  The model fits all possible ATA models to the data, then chooses the best model using the accuracy measures. Aggregate the best selected ATA forecast model for trend + remainder components and seasonal component to generate the final result. The final outcome is calculated from the forecasts from the single ATA models.
- Step 6 Inverse Transformation

***Second Algorithm* in Figure 1b**

- Step 1 Seasonality Test
- Step 2 Decomposition
- Step 3 Transformation
- Step 4 ATA Forecasting
- Step 5 Selection
- Step 6 Inverse Transformation and Aggregation

To summarize, the **ATAforecasting** procedure is given in Figure 1. As default, initially, the selected power family transformation is implemented, and the series are decomposed into the seasonal part and trend + remainder part, using the selected decomposition technique. Then, the Ata method is applied to the trend + remainder part. The components are added together again, and the selected power family transformation is inverted.


Enter Time Series Data
Transformation for stabilizing the variance of a time series?
Yes
No
Transformed data
Seasonality? (default for unit root: ADF test) (default for seasonality: autocorrelogram)
Decomposition (default:"classical decomp") (alternatively: stl, stlplus, sTR, tbats)
Deseasonalized data
ATAforecasting
Fit and select best model
Forecast
Aggregation
Inverse Transform

(a) First Algorithm **(b)** Second Algorithm

Figure 1: Algorithms of ATAforecasting procedure.

Power transformation family for ATAforecasting package

Traditional statistical procedures often assume that the data is homoscedastic and normally distributed. Despite its apparent restrictions, the logarithmic transformation has been used mostly when data violates these assumptions. The purpose of a particular transformation for better fitting is additivity, convergence to normality, stationarity, linearity, reduction of skewness, stabilizing variance. These purposes, which may even be inconsistent, are quite significant as it is just under such assumptions that particular statistical methods are relevant. Generally, logarithmic transformations almost stabilize the variance for time series consisting of large values. Some of the problems that arise when implementing a specific transformation are argued in different settings by Sakia (1992), Staniswalis et al. (1993), Quiroz et al. (1996), Yeo and Johnson (2000), Chen et al. (2002), Mu and He (2007), Horowitz (2009) and Meintanis and Stupfler (2015). There are many proposed methods of transformation and a large amount of research in the literature. Sakia (1992) provided a detailed and extensive review of the Box–Cox (Box and Cox (1964)) and some alternative versions. Different methodology recommended for choosing the appropriate value of transformation parameters based on maximizing the likelihood function (Box and Cox (1964)) or alternatively, Kullback-Leibler information-based method (Hernandez and Johnson, 1980), robust adaptive method (Carroll, 1980) and a method based on Kendall's rank correlation, (Han, 1987).

A chiefly used algorithm of the Box–Cox family is the logarithm transformation, which is convenient for multiplicative process data. Moreover, the asymptotic variance of a time series can be stabilized by the log-transformation. A shift parameter was additionally proposed to apply the log transformations more responsive and handy. The parameterizations of the shift parameter depend on knowledge of the data e.g., data range, data distribution, so user intervention is usually required. **ATAforecasting** package automates the selection of shift parameter, which is an important contribution of automatic times series forecasting.

Selected transformation functions included in the **ATAforecasting** R package provide the applicability of different types of transformation techniques for the data to which the Ata method will be applied. The `ATA.Transform` function works with many different types of inputs. Many power transformation methods are available to stabilize linearity and variance. In this package, power transformation family is consist of "Box–Cox", "Sqrt", "Reciprocal", "Log", "NegLog", "Modulus", "Bickel–Doksum", "Manly", "Dual", "Yeo–Johnson", "GPower", "GLog". If the transformation process needs a shift parameter, `ATA.Transform` will calculate the required shift parameter automatically.

- *Log Transformation with Shift*,
- *Box–Cox Transformation with Shift* (Box and Cox, 1964),
- *GLog Transformation* (Durbin et al., 2002),
- *NegLog Transformation* (Whittaker et al., 2005),
- *Reciprocal Transformation* (Tukey, 1977),
- *Bickel–Doksum Transformation* (Bickel and Doksum, 1982),
- *Yeo–Johnson Transformation* (Yeo and Johnson, 2000),
- *Modulus Transformation* (John and Draper, 1980),
- *Dual Power Transformation* (Yang, 2005),
- *GPower Transformation* (Kelmansky et al., 2013),

The `ATA.BoxCoxAttr` function

Since the main `ATA` function and `ATA.Transform` are designed by some attributes of Box–Cox power transformation family, we provide the user with the function `ATA.BoxCoxAttr`.

The R function `ATA.BoxCoxAttr` can be utilized with the following code,

```
ATA.BoxCoxAttr(bcMethod = "loglik", bcLower = 0
, bcUpper = 1,  bcBiasAdj = FALSE),
```

and makes use of four parameters. These are

- *bcMethod*: Choose method to be used in calculating lambda. "loglik" is default. Other method is "guerrero" (Guerrero, 1993).
- *bcLower*: Lower limit for possible lambda values. The lower value is limited by -5. Default value is 0.
- *bcUpper*: Upper limit for possible lambda values. The upper value is limited by 5. Default value is 1.
- *bcBiasAdj*: Use adjusted back-transformed mean for Box–Cox transformations. If transformed data is used to produce forecasts and fitted values, a regular back transformation will result in median forecasts. If *bcBiasAdj* is TRUE, an adjustment will be made to produce mean forecasts and fitted values. If *bcBiasAdj=TRUE*, optional parameter *fvar* required. *fvar* can either be the forecast variance or a list containing the interval *level* and the corresponding *upper* and *lower* intervals. Default value of *fvar* is NULL and it can't be changed.

The `ATA.Transform` function

The main function of power transformations, the `ATA.Transform`, can be called with

```
ATA.Transform(X, tMethod = "Box_Cox", tLambda
, tShift = 0, bcMethod = "loglik", bcLower = 0, bcUpper = 1)
```

and it makes use of seven parameters and returns three outputs. The inputs are

- *X*: a numeric vector or time series of class ts or msts for in-sample.
- *tMethod*: Power transformation family is consist of "Box_Cox", "Sqrt", "Reciprocal", "Log", "NegLog", "Modulus", "BickelDoksum", "Manly", "Dual", "YeoJohnson", "GPower", "GLog" in **ATAforecasting** package. If the transformation process needs shift parameter, `ATA.Transform` will calculate the required shift parameter automatically.
- *tLambda*: Box–Cox power transformation family parameter. If NULL, data transformed before model is estimated.
- *tShift*: Box–Cox power transformation family shifting parameter. If NULL, data transformed before model is estimated.
- *bcMethod*: Choose method to be used in calculating lambda. "loglik" is default. Other method is "guerrero" (Guerrero, 1993).
- *bcLower*: Lower limit for possible lambda values. The lower value is limited by -5. Default value is 0.
- *bcUpper*: Upper limit for possible lambda values. The upper value is limited by 5. Default value is 5.

The outputs are

- *trfmX* : Transformed data.
- *tLambda*: Box–Cox power transformation family parameter.
- *tShift* : Box–Cox power transformation family shifting parameter.

To apply this algorithm to an example in the **tsibbledata** package "aus_retail", monthly retail turnover (in million AUD) in Australian states from April 1982 to December 2018, we use the following commands.

```
library(tsibbledata)
library(lubridate)
library(dplyr)
library(tsbox)
library(ATAforecasting)
train_data <- aus_retail %>% filter(State == "New South Wales"
  , Industry == "Department stores"
  , `Series ID`== "A3349790V")
train_data <- tsbox::ts_ts(train_data)
bc_attr_set <- ATA.BoxCoxAttr(bcMethod = "loglik", bcLower = 0, bcUpper = 1)
fit_bc <- ATA(train_data, seasonal.type = "M", model.type = "A"
, seasonal.test = TRUE, seasonal.model = "decomp", plot.out = TRUE
, transform.method = "Box_Cox", transform.order = "before"
, transform.attr = bc_attr_set, negative.forecast = FALSE)
```

Seasonality for ATAforecasting package

Seasonality is a well-known phenomenon observed in many economic time series. Seasonal decomposition, which is the first stage of a time series modeling, is also the first stage of the Ata method. The performance of the Ata method has been improved after the seasonal decomposition.

Specifically, our proposed methodology to identify seasonality in time series is as follows. After or before implementing a Box—Cox transformation (if necessary) to the data, the data is decomposed into remainder, seasonal, and trend components. The trend and remainder components are then forecasted via the Ata method, the seasonal component is added back in, and the Box—Cox transformation is inverted. Then, point forecasts are calculated using each of the different models, and/or the resulting forecasts are able to be combined.

In previous studies, the classical decomposition method is much used after the seasonality test. With this package, stl, stlplus, tbats, and stR decomposition techniques are also available choices by the **ATAforecasting** package, which can be chosen with only one or multiple.

Seasonality for **ATAforecasting** package enables estimating all of the below components and specifications. The main functions of seasonality in the package are the following

- `ATA.SeasAttr()`,
- `ATA.Seasonality()`,
- `ATA.Decomposition()`.

Three seasonality diagnostics methods are able to be applied in the package.

- Unit Root Tests,
- Seasonal Unit Root Tests,
- Seasonal Decomposition.

The `ATA.SeasAttr` function

This function is a class of seasonality tests using `corrgram.test` from **ATAforecasting** package, `ndiffs` and `nsdiffs` functions from **forecast** package. Also, `ndiffs` and `nsdiffs` functions have been modified according to different unit root testing packages. Please review manual and vignette documents of the latest **forecast** package. According to **forecast** package, `ndiffs` and `nsdiffs` functions estimate the number of differences requisite to ensure stationary of a given time series.

`ndiffs` employs unit root tests to define required number of differences for time series to be ensured trend stationary. `nsdiffs` employs seasonal unit root tests to define required number of seasonal differences for time series to be ensured trend stationary.

The `ATA.SeasAttr` function works with many different types of inputs. The inputs are below.

- *corrgram.tcrit*: *t*-value for periodogram seasonality test.
- *uroot.test*: Type of unit root test before all type seasonality test. Possible values are "adf", "pp", and "kpss".
- *suroot.test*: Type of seasonal unit root test to use. Possible values are "correlogram", "sea", "hegy", "ch", and "ocsb".
- *suroot.uroot*: If TRUE, unit root test for stationary before seasonal unit root test is allowed.
- *uroot.type*: Specification of the deterministic component in the regression for unit root test. Possible values are "level" and "trend".
- *uroot.alpha*: Significant level of the unit root test, possible values range from 0.01 to 0.1.
- *suroot.alpha*: Significant level of the seasonal unit root test, possible values range from 0.01 to 0.1.
- *uroot.maxd*: Maximum number of nonseasonal differences allowed.
- *suroot.maxD*: Maximum number of seasonal differences allowed.
- *suroot.m*: Deprecated. Length of seasonal period: frequency of data for `nsdiffs`.
- *uroot.pkg*: Using **ucra** or **tseries** packages for unit root test. The default value is **ucra**.
- *multi.period*: Selection type of multiseasonal period. *min* or *max* function for selection.
- *x13.estimate.maxiter Maximum*: iteration for X13ARIMA/SEATS estimation.
- *x13.estimate.tol*: Convergence tolerence for X13ARIMA/SEATS estimation.
- *x11.estimate.maxiter Maximum*: iteration for X11 estimation.
- *x11.estimate.tol*: Convergence tolerence for X11 estimation.

Unit root tests

Unit root tests for stationarity have compatibility in almost every practical time series analysis. Choosing which unit root procedure to employ is an issue of active interest. In this study, we implement the three widely used unit root tests. In accordance with past research, the selected unit root tests occasionally disagree in choosing the convenient order of integration for a given data. The following literature shows the basic features of unit root tests. Users who demand details should consult the original resources and standard references (see, for example, (Davidson and MacKinnon, 1993; Hamilton, 1994; Hayashi, 2000)).

In the **ATAforecasting** package, the following unit roots methods are able to be applied.

- *Augmented Dickey–Fuller Test (Dickey and Fuller, 1979; Said and Dickey, 1984)*
- *Phillips and Perron Test (Phillips and Perron, 1988)*
- *Kwiatkowski, Phillips, Schmidt, and Shin Test (Kwiatkowski et al., 1992)*

The `ATA.SeasAttr` function for unit root tests

Since the main `ATA` function and `ATA.Seasonality` are designed by some attributes of unit root tests, we provide the user with the function `ATA.SeasAttr`.

For the main function `ATA`, the attributes of unit root test can be accessed with

```
ATA.SeasAttr(uroot.pkg = "tseries", uroot.test = "kpss"
 , uroot.type = "trend", uroot.alpha = 0.05)
```

The following code uses the unit root test approach to search trend component before the seasonality test of the data in the context of the Ata method.

```
seas_attr_set <- ATA.SeasAttr(suroot.test = "correlogram"
, corrgram.tcrit = 1.28, uroot.pkg = "tseries"
, uroot.test = "kpss", uroot.type = "trend"
, uroot.alpha = 0.05)

fit_seas <- ATA(train_data, model.type = "A", seasonal.type = "M"
, seasonal.test = TRUE, seasonal.model = "tbats", plot.out = TRUE
, seasonal.test.attr = seas_attr_set, negative.forecast = FALSE)
```

Seasonality tests

There are numerous studies relating seasonality and unit root. One of these studies uses autocorrelogram. Autocorrelation function and partial autocorrelation function are useful qualitative tools to estimate the existence of autocorrelation at individual lags. The Ljung-Box Q-test is a more quantitative method to test autocorrelation at multiple lags jointly. Other techniques generally use unit root tests. Hylleberg et al. (1990) improved unit root tests in linear time series regarding seasonality and studied with different models, including different combinations of seasonal, trend, remainder, and constant parts. Their purpose is to improve a testing process that will determine what class of seasonality is accountable for the seasonality in a time series process. There exist more studies for testing seasonal unit roots, such as Ljung and Box (1978), Dickey et al. (1984), Osborn et al. (1988), and Wang et al. (2006).

In the **ATAforecasting** package, the following methods are able to applied.

- *Autocorrelogram (Ljung and Box, 1978)*
- *Canova–Hansen (CH) Test (Canova and Hansen, 1995)*
- *Hylleberg, Engle, Granger & Yoo (HEGY) Test (Hylleberg et al., 1990)*
- *Osborn, Chui, Smith, Birchenhall (OCSB)Test (Osborn et al., 1988)*
- *Seasonal Strength Measure (Wang et al., 2006)*

The `ATA.SeasAttr` function for seasonal unit root test

Since the main `ATA` function and `ATA.Seasonality` are designed by some attributes of seasonality tests, we provide the user with the function `ATA.SeasAttr`.

For the main function `ATA`, the attributes of seasonality test can be accessed with

```
seas_attr_set <- ATA.SeasAttr(suroot.test = "correlogram"
  , corrgram.tcrit = 1.28)

seas_attr_set <- ATA.SeasAttr(suroot.test = "ocsb", suroot.alpha = 0.05)
```

An example of the seasonality test's call is the following

```
seas_attr_set <- ATA.SeasAttr(suroot.test = "ocsb", suroot.alpha = 0.05
, uroot.pkg = "tseries", uroot.test = "adf", uroot.type = "trend"
, uroot.alpha = 0.05)

fit_seas <- ATA(train_data, model.type = "A", seasonal.type = "M"
, seasonal.test = TRUE, seasonal.model = "stl", plot.out = TRUE
, seasonal.test.attr = seas_attr_set, negative.forecast = FALSE)
```

The `ATA.Seasonality` function for Seasonality Tests

The **ATAforecasting**'s seasonality diagnostics described before in this paper are implemented into a function named `ATA.Seasonality` that can calculate all of them respectively. The function syntax is

```
ATA.Seasonality(input = train_data, ppy = frequency(train_data)
  , attr_set = seas_attr_set)
```

The `ATA.Seasonality` function works with many different types of inputs. The inputs are below.

- *input*: The data.
- *ppy*: Frequency of the data.
- *attr_set*: Assign from `ATA.SeasAttr` function. Attributes set for the unit root, seasonality tests.

Here is an another simple example, applying `ATA.SeasAttr` and `ATA.Seasonality` to the M3 data:

```
library(ATAforecasting)
library(Mcomp)
seas_attr_set <- ATA.SeasAttr(suroot.test = "correlogram"
  , corrgram.tcrit = 1.28, uroot.pkg="tseries"
  , uroot.test="adf", uroot.type = "trend"
  , uroot.alpha = 0.05, uroot.maxd = 1)

is.season <- ATA.Seasonality(M3[[1899]]$x
  , frequency(M3[[1899]]$x)
  , seas_attr_set)
```

Seasonal decomposition

A substantial aim in time series analysis is the decomposition of a time series into latent parts that can be incorporated with dissimilar versions of temporal variations. Persons (1919) was the first to state the assumptions of latent parts particularly. Persons indicated that time series was constituted of four types of fluctuations (Dagum and Bianconcini, 2016): *residual variations*, *seasonal movement*, *secular trend*, and *cyclical movements*. Further, research in that direction included Copeland (1915) and Persons (1919), who introduced for extracting the seasonal component until Macaulay (1931) suggested a technique which turned into "classical" in the log run.

Macaulay (1931) developed a computer program which is significantly simplified the calculations (Shiskin, 1957). Extensively used various techniques and features which have such as ARIMA extensions, regressors, calendar effects, robustness, and extensive diagnostics in literature are X-11 (Shiskin et al., 1967), X-11-ARIMA (Dagum, 1988), X-12-ARIMA (Findley et al., 1998) and X-13ARIMA-SEATS (Monsell et al., 2003; Findley, 2005; Monsell, 2007). X-13ARIMA-SEATS contains a type of the TRAMO/SEATS procedure which was improved by the Bank of Spain for seasonal adjustment.

Cleveland et al. (1990) recommended a different approach and developed STL (Seasonal Trend decomposition using LOESS) based on local regression familiar as moving regression which is a generalization of moving average and polynomial regression. LOESS is a connected nonparametric method that assembles multiple regression models in a metamodel based on the k-nearest neighbor. Burman (1980) discussed plenty of seasonal adjustment techniques and remarked that all but one were ad hoc techniques. Since this study, several model-based methods for seasonal decomposition have been evolved, including the TRAMO/SEATS procedure, assorted structural time series models (Harvey, 1990; Commandeur et al., 2011) and the BATS and TBATS models of Livera et al. (2011).

Conventionally, the four variations suppose to be mutually independent of one another and signify by means of an additive decomposition model. If there is dependency among the hidden parts, this relation is signified via a multiplicative decomposition model. In some cases, combined additive and multiplicative models can be employed. See Dagum and Bianconcini (2016) for further details.

The `ATA.Decomposition` function for seasonality

Automatic seasonal decomposition for the ATA method is called `ATA.Decomposition` function in the **ATAforecasting** package. The function returns seasonally adjusted data constructed by removing the seasonal component. The methodology is fully automatic. The `ATA.Decomposition` function works with many different types of inputs. The inputs are below.

- *input*: It must be `ts`, `msts`, or `numeric` object. If it is a `numeric` object, `findPeriod` must be 1, 2, 3 or 4. If it is a `msts` object, `findPeriod` must be 3 or 4.
- *s.model*: A string identifying method for seasonal decomposition. If NULL, "decomp" method is default. c("none", "decomp", "stl", "stlplus", "tbats", "stR") phrases of methods denote.
  - none: seasonal decomposition is not required.
  - decomp: classical seasonal decomposition. If *decomp*, the **stats** package will be used.
  - stl: seasonal-trend decomposition procedure based on LOESS developed by Cleveland et al. (1990) . If *stl*, the **stats** and **forecast** packages will be used. Multiple seasonal periods are allowed.
  - stlplus: seasonal-trend decomposition procedure based on LOESS developed by Cleveland et al. (1990). If *stlplus*, the **stlplus** package will be used.
  - tbats: exponential smoothing state space model with Box–Cox transformation, ARMA errors, trend, and seasonal components as described in Livera et al. (2011). Parallel processing is used by default to speed up the computations. If *tbats*, the **forecast** package will be used. Multiple seasonal periods are allowed.
  - stR: seasonal-trend decomposition procedure based on the regression developed by Dokumentov and Hyndman (2015). If *stR*, the **stR** package will be used. Multiple seasonal periods are allowed.
  - x13: seasonal-trend decomposition procedure based on X13ARIMA/SEATS. If *x13*, the **seasonal** package will be used.
  - x11: seasonal-trend decomposition procedure based on X11. If *x11*, the **seasonal** package will be used.
- *s.type*: A one-character string identifying method for the seasonal component framework. If NULL, "M" is default. The letter "A" for additive model, the letter "M" for multiplicative model.
- *s.frequency*: Value(s) of seasonal periodicity. If *s.frequency* is not integer, *X* must be `msts` time series object. c(s1,s2,s3,...) for multiple period. If *X* has multiple periodicity, "tbats" or "stR"

seasonal model have to be selected. For example, period of the input data, which have one seasonal pattern –> 12 for monthly / 4 for quarterly / 7 for daily / 5 for business days. Periods of the input data which have complex/multiple seasonal patterns –> c(7,354.37,365.25).

- *seas_attr_set*: Assign from `ATA.SeasAttr` function. Attributes set for unit root, seasonality tests.

`ATA.Decomposition` function returns four outputs. The outputs are below.

- *AdjustedX* : Deseasonalized data.
- *SeasIndex* : Particular seasonality data given cycle/frequency.
- *SeasActual* : Seasonality given original data.
- *SeasType* : Seasonal decomposition technique.

As an example, let us compute seasonal decomposition on the real life **tsibbledata** dataset shown in the following seven figures (Figures 2, 3, 4, 5, 6, and 7).

```
best_fit_seas <- ATA(train_data, start.phi = 0.80, end.phi = 0.99
, size.phi = 0.01, train_test_split = 18, seasonal.test = TRUE
, seasonal.model = c("decomp","stl", "stlplus","tbats", "stR")
, negative.forecast = FALSE, plot.out = TRUE)
best_fit_seas$is.season
```

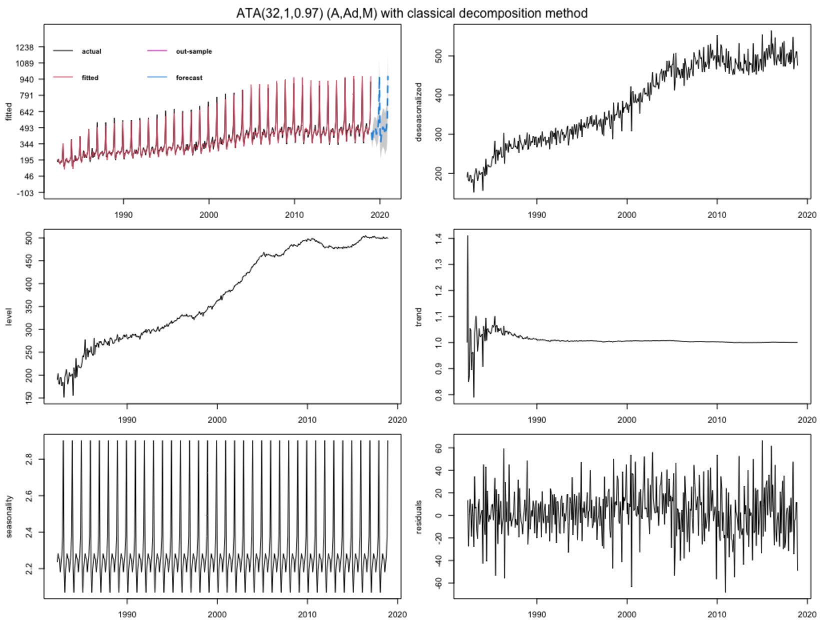


Figure 2: Best forecasts for the Australian Retail Turnover using ATA seasonal trended methods.

```
library(ggplot2)
autoplot(train_data) +
autolayer(fit_decomp$forecast, series="ATA-decomp") +
autolayer(fit_stl$forecast, series="ATA-stl") +
autolayer(fit_stlplus$forecast, series="ATA-stlplus") +
autolayer(fit_stR$forecast, series="ATA-stR") +
autolayer(fit_tbats$forecast, series="ATA-tbats") +
ggtitle("Forecasts from ATA seasonal trended methods") + xlab("Year") +
ylab("Monthly Retail Trade Turnover of Australian States") +
guides(colour=guide_legend(title="Forecast"))
```

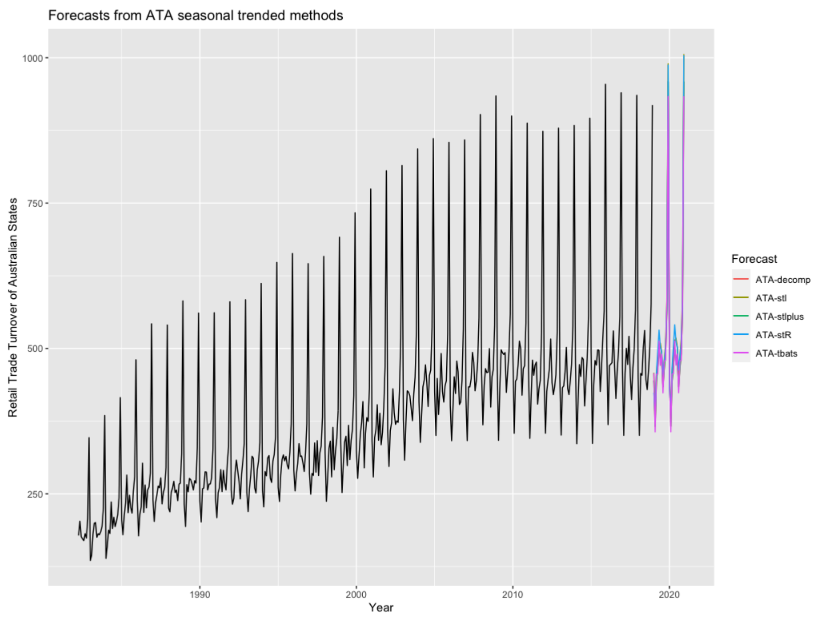


Figure 3: Forecasts from ATA seasonal trended methods.

There are five different techniques for seasonal decomposition in the package. We use the following techniques

- *Classical Decomposition*: The classical method of time series decomposition originated in the 1920s and was widely used until the 1950s.

```
fit_decomp <- ATA(train_data, seasonal.test = TRUE
  , seasonal.model = "decomp" , negative.forecast = FALSE)
```

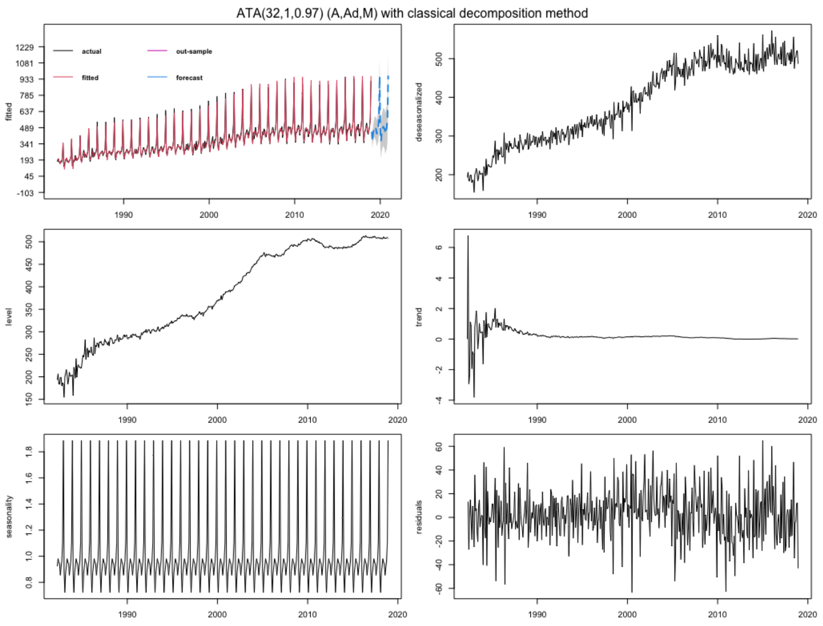


Figure 4: The Ata method with classical decomposition.

- *STL Decomposition* (Cleveland et al., 1990):

```
fit_stl <- ATA(train_data,  model.type = "A", seasonal.type = "M"
, seasonal.test = TRUE, seasonal.model = "stl", negative.forecast = FALSE)
```

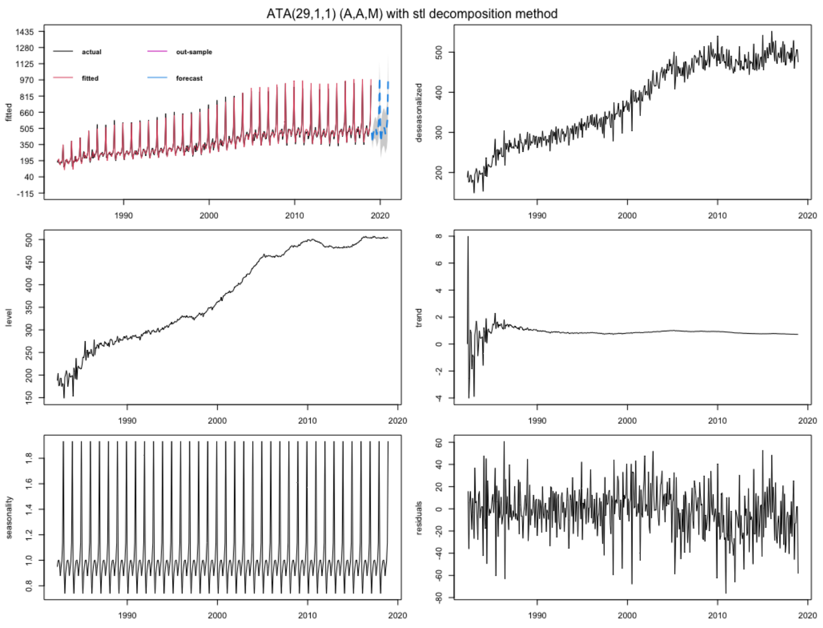


Figure 5: The Ata method with STL decomposition.

- *STL+ Decomposition (Hafen, 2010)*: The STL+ is implemented in **stlplus** R package. See more details in Hafen (2010).

```
fit_stlplus <- ATA(train_data, model.type = "A", seasonal.type = "M"
, seasonal.test = TRUE, seasonal.model = "stlplus", negative.forecast = FALSE)
```

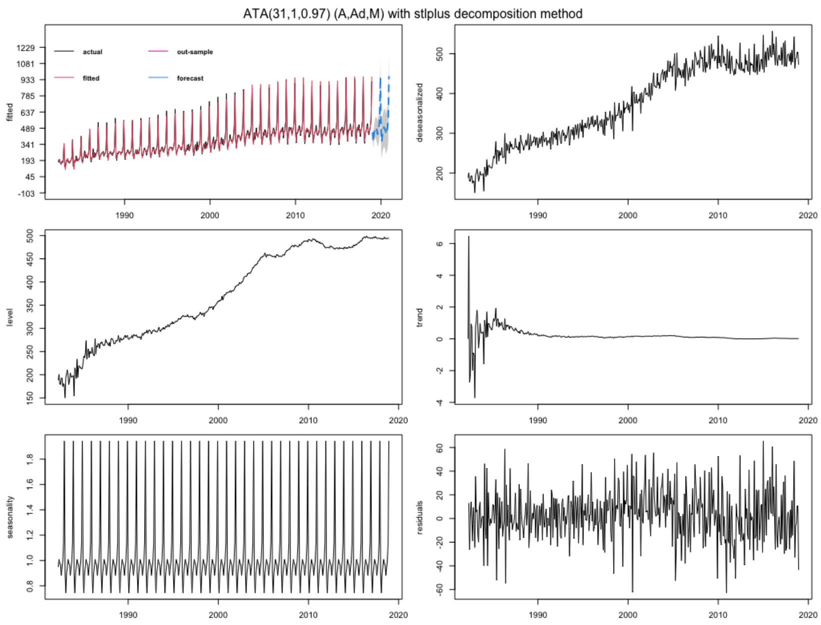


Figure 6: The Ata method with STLplus decomposition.

- *TBATS Decomposition (Livera et al., 2011)*:TBATS uses Box–Cox transformation, exponential smoothing, trigonometric seasonality and ARMA errors (Livera et al., 2011).

```
fit_tbats <- ATA(train_data, seasonal.test = TRUE, seasonal.model = "tbats"
, level.fixed = TRUE, negative.forecast = FALSE, plot.out = TRUE)
```

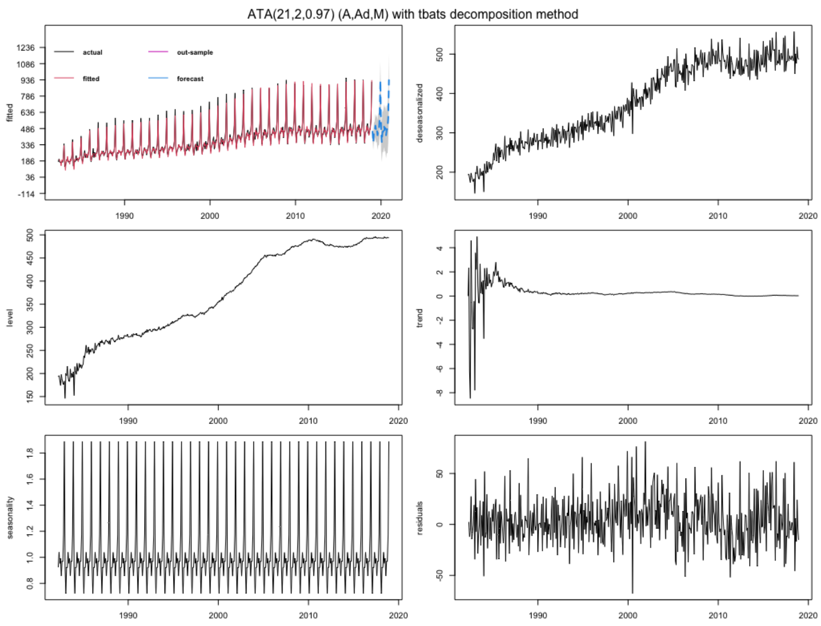


Figure 7: The Ata method with TBATS decomposition.

- *stR Decomposition* (Dokumentov and Hyndman, 2015): Seasonal-Trend decomposition procedure based on Regression (stR) is similar to Ridge Regression, and *Robust stR* can be related to LASSO. The stR procedure grants for multiple seasonal and cyclic components and multiple linear regressors with constant, flexible, seasonal, and cyclic effect. The Seasonal-Trend decomposition by Regression is implemented in **stR** R package.

```
fit_stR <- ATA(train_data, seasonal.test = TRUE, seasonal.model = "stR"
, negative.forecast = FALSE, plot.out = TRUE)
```

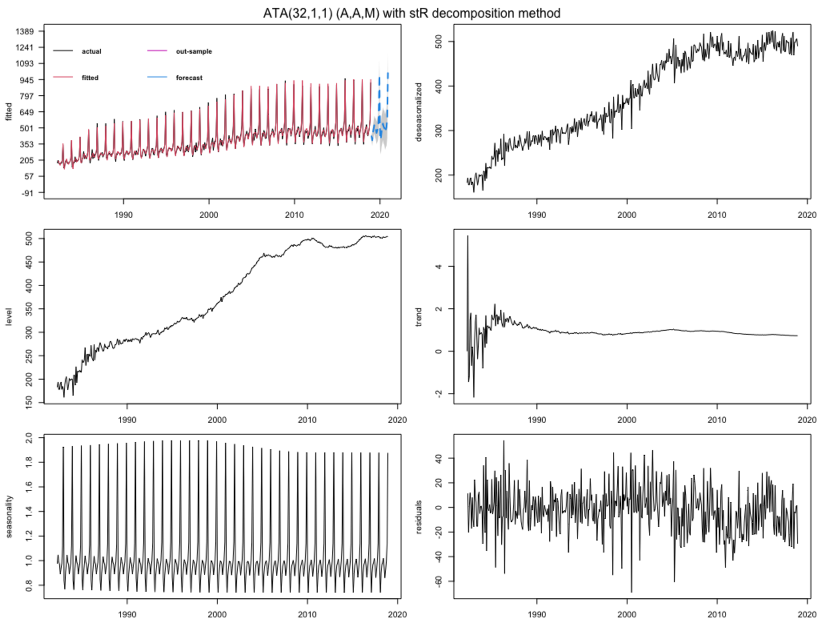


Figure 8: The Ata method with stR decomposition.

Univariate time series forecasting with the Ata method

Ata method is an innovative new forecasting technique where the forms of the models are similar to ES models. Still, the smoothing parameters depend on the sample size, are optimized on a discrete space. Initialization is both easier as it is done simultaneously when the parameters are optimized and is less influential since the weights assigned to initial values approach zero quickly. The Ata method can easily be applied to all time series settings and provides better forecasting performance due to its flexibility. ATA-damped is a version of the Ata method that mainly focuses on the trend component, allowing it to range both in magnitude and form.

For a time series $\{y_1, \ldots, y_n\}$, the Ata method can be given in additive form as below:

$$l_t = \left(\frac{p}{t}\right) y_t + \left(\frac{t-p}{t}\right)(l_{t-1} + \phi b_{t-1}), \tag{1}$$

$$b_t = \left(\frac{q}{t}\right)(l_t - l_{t-1}) + \left(\frac{t-q}{t}\right)(\phi b_{t-1}), \tag{2}$$

where p is the smoothing parameter for level, q is the smoothing parameter for trend, ϕ is the dampening parameter and $l_t = y_t$ for $t \leq p$, $b_t = y_t - y_{t-1}$ for $t \leq q$, $b_1 = 0$, $p \in \{1, 2, \ldots, n\}$, $q \in \{0, 1, 2, \ldots, p\}$, $\phi \in (0, 1]$, and $p \geq q$. Then, the h step ahead forecasts can be obtained by:

$$\hat{y}_{t+h|t} = l_t + \left(\phi + \phi^2 + \ldots + \phi^h\right) b_t. \tag{3}$$

Similarly for a time series $\{y_1, \ldots, y_n\}$, the Ata method can be given in multiplicative form as below:

$$l_t = \left(\frac{p}{t}\right) y_t + \left(\frac{t-p}{t}\right)\left(l_{t-1} b_{t-1}^{\phi}\right), \tag{4}$$

$$b_t = \left(\frac{q}{t}\right)\left(\frac{l_t}{l_{t-1}}\right) + \left(\frac{t-q}{t}\right)\left(b_{t-1}^{\phi}\right), \tag{5}$$

where, again, p is the smoothing parameter for level, q is the smoothing parameter for trend, ϕ is the dampening parameter and $l_t = y_t$ for $t \leq p$, $b_t = \frac{y_t}{y_{t-1}}$ for $t \leq q$, $b_1 = 1$, $p \in \{1, 2, \ldots, n\}$, $q \in \{0, 1, 2, \ldots, p\}$, $\phi \in (0, 1]$, and $p \geq q$. Then, the h step ahead forecasts can be obtained by:

$$\hat{y}_{t+h|t} = l_t + b_t^{(\phi+\phi^2+\ldots+\phi^h)}. \tag{6}$$

Since both versions of the method require three parameters, we will distinguish between them by using the notation $ATA_{add}(p, q, \phi)$ for the additive form and $ATA_{mult}(p, q, \phi)$ for the multiplicative form.

Notice that when $q = 0$, both forms of ATA are reduced to the simple form $ATA(p, 0, \phi)$ which can be written as:

$$l_t = \left(\frac{p}{t}\right) y_t + \left(\frac{t-p}{t}\right) l_{t-1}, \tag{7}$$

where $p \in \{1, 2, \ldots, n\}$ and $l_t = y_t$ for $t \leq p$. Forecasts then can be obtained by $\hat{y}_{t+h|t} = l_t$.

When $q \neq 0$ and $\phi = 1$, the additive and multiplicative forms of ATA are reduced to the trended versions $ATA_{add}(p, q, 1)$ and $ATA_{mult}(p, q, 1)$, which are given below, respectively:

$$l_t = \left(\frac{p}{t}\right) y_t + \left(\frac{t-p}{t}\right) (l_{t-1} + b_{t-1}), \tag{8}$$

$$b_t = \left(\frac{q}{t}\right) (l_t - l_{t-1}) + \left(\frac{t-q}{t}\right) (b_{t-1}), \tag{9}$$

$$\hat{y}_{t+h|t} = l_t + hb_t, \tag{10}$$

and

$$l_t = \left(\frac{p}{t}\right) y_t + \left(\frac{t-p}{t}\right) (l_{t-1}b_{t-1}), \tag{11}$$

$$b_t = \left(\frac{q}{t}\right)\left(\frac{l_t}{l_{t-1}}\right) + \left(\frac{t-q}{t}\right) (b_{t-1}), \tag{12}$$

$$\hat{y}_{t+h|t} = l_t + b_t^h. \tag{13}$$

To sum up, ATA can be given in 7 forms, namely the additive damped form $ATA_{add}(p, q, \phi)$ (equations (1-3)), multiplicative damped form $ATA_{mult}(p, q, \phi)$ (equations (4-6)), simple form $ATA(p, 0, \phi)$ (equation (7)), additive trend form $ATA_{add}(p, q, 1)$ (equations (8-10)), and finally multiplicative trend form $ATA_{mult}(p, q, 1)$ (equations (11-13)).

Another distinction can be made based on the parameter optimization process used for these forms. Unless otherwise stated, the parameter values that minimized the in-sample one step ahead using selected accuracy measures such as sMAPE, MASE, or OWA are used as optimum values, and optimization is carried out for all the parameters simultaneously. However, in some cases, we realized that fixing the smoothing parameter for the level and then optimizing the trend parameter can be beneficial. We call these the "level-fixed" versions of ATA. The optimization is carried out for these models as follows:

1. Find the value of p that minimized the in-sample one step ahead $sMAPE$ for $q = 0$ and $\phi = 1$. Call this value p^*.
2. Holding $p = p^*$ fixed optimize q (and ϕ if needed) ,again, by minimizing the in-sample one step ahead $sMAPE$.

Models where the parameter optimization is carried out using the algorithm in 1. and 2. will receive the superscript (lf) an abbreviation for "level-fixed" such as $ATA_{add}^{lf}(p, q, \phi)$ or $ATA_{mult}^{lf}(p, q, \phi)$.

Obtaining prediction interval

For forecasting horizon h, the prediction interval is obtained by:

$$y_{n+h|n} \pm C_h, \tag{14}$$

where $C_h = \sqrt{h} Z_{\alpha/2} S_e$, $Z_{\alpha/2}$ is the Normal deviate corresponding to $(1-\alpha)\%$ confidence interval, and S_e is the standard deviation of the one step ahead errors of model fitting. If any lower bounds are found to be negative, they are set equal to zero.

Model selection

There are plenty of measures and criteria available in the forecasting literature for interpreting the achievements and accuracy of forecasting methods. In the M-Competitions, some of these measures were employed without any obvious consensus as to the pros and cons of each.

A forecast error is a difference between an observed value and its forecast. Forecast errors are different from residuals in two aspects. Firstly, residuals are computed on the training set, while forecast errors are computed on the test set. Secondly, residuals are based on one-step-ahead forecasts, while forecast errors can contain multi-step forecasts (Hyndman and Athanasopoulos, 2019).

Let Y_t indicates the observation at time t, and F_t indicates the forecast of Y_t. The forecast error $e_t = Y_t - F_t$ is calculated. The forecasts are calculated from a common base time and are of varying forecast horizons. Hence, we calculate out of sample forecasts $F_{n+1}, \ldots, \quad F_{n+m}$ based on data from times $t = 1, \ldots, n$. Optionally, the forecasts can be from varying base times and be of a coherent forecast horizon. Namely, we can calculate forecasts $F_{1+h}, \ldots, F_{m+h}$ where each F_{j+h} is based on data from times $t = 1, \ldots, n$. The in-sample forecasts in the examples above were based on the second scenario with $h = 1$. A third scenario shows up when we request to compare the accuracy of methods across many series at a forecast horizon. Then we calculate a single F_{n+h} based on data from times $t = 1, \ldots, n$ for each of m different series (Hyndman and Koehler, 2006). In this study, we adapt M4 and prior M-Competitions' accuracy measures pool.

Automatic forecasting

We unite the prior concepts to obtain a robust and widely appropriate automatic forecasting algorithm. The concept is summarized below.

1. Identify and correct for seasonality in time series, respectively.
   - Detect stationarity and seasonality in time series.
   - Decompose time series.
2. For the selected time series data, apply all models that are applicable, optimizing the parameters of the ATA model in each case.
3. Select the best of the ATA models according to the selected accuracy measure (SMAPE is default for the **ATAforecasting** package).
4. Generate point forecasts using the best model (with optimized parameters).
5. Obtain prediction intervals for the best model.

ATAforecasting in practice

This section introduces an overview of how the package is structured.

This software enables both numerical and graphical outputs to be displayed for all methods described in the previous section. This software is intended to be used with the R statistical program (R Core Team, 2016). Our package is composed of 13 functions that allow users to obtain estimates for all proposed methods. Details on the usage of the functions (described in Table 1) can be obtained with the corresponding help pages.

Returns ATA(p,q,ϕ) applied to X, based on the modified simple ES as described in Yapar (2018). The Ata method is a new univariate time series forecasting method that provides innovative solutions to issues faced during the initialization and optimization stages of existing methods. The ATA's forecasting performance is superior to existing methods both in terms of easy implementation and accurate forecasting. It can be applied to non-seasonal or deseasonalized time series, where the deseasonalization can be performed via any preferred decomposition method. This methodology performed extremely well on the M3 and M4-Competition data.

Functions of ATAforecasting package

Many functions, including `ATA`, `ATA.Forecast`, `ATA.Plot`, `ATA.Print`, `ATA.Accuracy`, `ATA.Seasonality`, `ATA.Transform`, `ATA.BackTransform` produce output in the form of a **ATAforecasting** object (i.e., an object of class "ata"). This package needs some R packages for unit root tests, seasonal unit root tests, seasonal decompositions, M3 dataset, M4 dataset, and benchmark forecast models to work consistently across a range of forecasting models. These R package are **Rcpp** (Eddelbuettel et al., 2020a), **RcppArmadillo** (Eddelbuettel et al., 2020b), **tseries** (Trapletti and Hornik, 2020), **forecast** (Hyndman et al., 2020), **urca** (Pfaff et al., 2016), **uroot** (de Lacalle, 2020), **seasonal** (Sax and Eddelbuettel, 2020), **stR** (Dokumentov and Hyndman, 2018), **stlplus** (Hafen, 2016), **xts** (Ryan et al., 2020), **timeSeries** (Wuertz et al., 2020), **TSA** (Chan and Ripley, 2020), **Mcomp** (Hyndman et al., 2018), **M4Comp2018** (BenTaieb, 2018).

Objects of class "ata" contain information about the forecasting method, the data used, the point forecasts obtained, prediction intervals, residuals, and fitted values. There are several functions designed to work with these objects, including `ATA.Forecast`, `ATA.Accuracy`, `ATA.Plot` and `ATA.Print`.

Description of the `ATA` function

`ATA` function produces a `ata` object directly. If the first argument is of class `ts` (time series object) or `msts` (multi seasonal time series objects), it returns forecasts from the automatic ATA algorithm discussed in this chapter. The definition of `ATA` function is below.

```
ATA(X, Y = NULL, parP = NULL, parQ = NULL, parPHI = NULL
, start.phi = NULL, end.phi = NULL, size.phi = NULL
, model.type = NULL, seasonal.test = NULL, seasonal.model = NULL
, seasonal.period = NULL, seasonal.type = NULL, find.period = NULL
, seasonal.test.attr = NULL, accuracy.type = NULL
, level.fixed = FALSE, trend.fixed = FALSE, trend.search = FALSE
, initial.level = NULL, initial.trend = NULL, h = NULL
, train_test_split = NULL, holdout = FALSE
, holdout.adjustedP = TRUE, holdout.set_size = NULL
, transform.order = "before", transform.method = NULL
, transform.attr = NULL, lambda = NULL, shift = NULL
, ci.level = 95, negative.forecast = TRUE
, plot.out = TRUE, print.out = TRUE)
```

Inputs of `ATA` function

The `ATA` function works with many different types of inputs. It generally takes a time series data or time series model as its main argument, and produces forecasts appropriately. It always returns objects of class "ata".

If the first argument is of class `ts` or `msts`, it returns forecasts from the automatic ATA algorithm discussed in this chapter before.

- ***X*** **:** A numeric vector or time series of class `ts` or `msts` for in-sample (trarining set).
- ***Y*** **:** A numeric vector or time series of class `ts` or `msts` for out-sample (test set). If you do not have out-sample data, you can split in-sample data into training and test dataset with *train_test_split* argument.
- ***h*****:** The number of steps to forecast ahead. When the parameter is NULL; if the frequency of *X* is 4 the parameter is set to 8; if the frequency of *X* is 5, the parameter is set to 10; if the frequency of *X* is 12, the parameter is set to 24; if the frequency of *X* is 24, the parameter is set to 48; the parameter is set to 6 for other cases.
- ***train_test_split*** **:** If *Y* is NULL, this parameter divides *X* into two parts: training set (in-sample) and test set (out-sample). *train_test_split* is number of periods for forecasting and size of test set. If the value is between 0 and 1, percentage of length is active.
- ***ci.level*** **:** Confidence Interval levels for forecasting.
- ***negative.forecast*** **:** Negative values are allowed for forecasting. Default value is TRUE. If FALSE, all negative values for forecasting are set to 0.
- ***plot.out*** **:** Default is TRUE. If FALSE, graphics of Ata method are not shown.
- ***print.out*** **:** Default is TRUE. If FALSE, summary of Ata method is not shown.

Level Parameters **:**

- ***parP*** **:** Value of Level parameter *p*. If NULL or "opt", it is estimated. *p* has all integer values from 1 to *length(X)*.
- ***level.fixed*** **:** If TRUE, *pStarQ* is selected. First, fits ATA(p,0) where p = p* is optimized for q=0. Then, fits ATA(p*,q) where q is optimized for p = p*.
- ***initial.level*** **:** If NULL, FALSE is default. If FALSE, Ata method calculates the *p*th observation in *X* for level. If TRUE, Ata method calculates average of first *p* value in *X* for level.

Trend Parameters **:**

- ***parQ*** **:** Value of Trend parameter q. If NULL or "opt", it is estimated. q has all integer values from 0 to p.
- ***parPHI*** **:** Value of Damping Trend parameter ϕ. If NULL or "opt", it is estimated. ϕ has all values from 0 to 1.
- ***model.type*** **:** An one-character string identifying method using the framework terminology. The letter "A" for additive model, the letter "M" for multiplicative model. If NULL, both letters will be tried and the best model (according to the accuracy measure *accuracy.type*) returned.
- ***initial.trend*** **:** If NULL, FALSE is default. If FALSE, Ata method calculates the qth observation in $X_T - X_{T-1}$ for trend. If TRUE, Ata method calculates average of first q value in $X_T - X_{T-1}$ for trend.
- ***trend.opt*** **:**
  1. ***none*** **:** none.
  2. ***fixed*** **:** *pBullet* is selected. Fits ATA(p,1) where p = p* is optimized for q = 1.
  3. ***search*** **:** *pBullet* is selected. Fits ATA(p,q) where p = p* is optimized for q = q* (q > 0). Then, fits ATA(p*,q) where q is optimized for p = p*.
- ***start.phi*** **:** Lower boundary for searching *parPHI*. If NULL, 0 is default.
- ***end.phi*** **:** Upper boundary for searching *parPHI*. If NULL, 1 is is default.
- ***size.phi*** **:** Increment step for searching `parPHI`. If NULL, the step size will be determined as the value that allows the bounds for the optimized value of `parPHI` to be divided into 20 equal parts.

Seasonal Parameters **:**

- ***seasonal.test*** **:** Testing for stationary and seasonality. If TRUE, the method firstly uses $test = "adf"$, Augmented Dickey-Fuller, unit-root test then the test returns the least number of differences required to pass the test at level α. After the unit-root test, a seasonal test applies on the stationary X.
- ***seasonal.type*** **:** A one-character string identifying method for the seasonal component framework. If NULL, "M" is default. The letter "A" for additive model, the letter "M" for multiplicative model. If other seasonal decomposition method except *decomp* with "M", Box–Cox transformation with *lambda*=0 is selected.
- ***seasonal.model*** **:** A string identifying method for seasonal decomposition. If NULL, "decomp" method is default. c("none", "decomp", "stl", "stlplus", "tbats", "stR") phrases of methods denote.
  - ***none*** **:** seasonal decomposition is not required.
  - ***decomp*** **:** classical seasonal decomposition. If *decomp*, the **stats** package will be used.
  - ***stl*** **:** seasonal-trend decomposition procedure based on LOESS developed by Cleveland et al. (1990). If *stl*, the **stats** package will be used.
  - ***stlplus*** **:** seasonal-trend decomposition procedure based on LOESS developed by Cleveland et al. (1990). If *stlplus*, the **stlplus** package will be used.
  - ***tbats*** **:** exponential smoothing state space model with Box–Cox transformation, ARMA errors, trend, and seasonal components as described in Livera et al. (2011). Parallel processing is used by default to speed up the computations. If *tbats*, the **forecast** package will be used.
  - ***stR*** **:** seasonal-trend decomposition procedure based on regression developed by Dokumentov and Hyndman (2015). If *stR*, the **stR** package will be used.
  - ***x13*** **:** seasonal-trend decomposition procedure based on X13ARIMA/SEATS. If ***x13***, the **seasonal** package will be used.
  - ***x11*** **:** seasonal-trend decomposition procedure based on X11. If ***x11***, the **seasonal** package will be used.
- ***seasonal.period*** **:** Value(s) of seasonal periodicity. If NULL, *frequency* of X is default If *seasonal.period* is not integer, *X* must be an `msts` time series object. c(s1, s2, s3,...) for multiple period. If *X* has multiple periodicity, "tbats" or "stR" seasonal model have to be selected.
- ***seasonal.test.attr*** **:** Attributes set for unit root, seasonal unit root test, seasonality tests and X13ARIMA/SEATS, and X11. If you want to change, please use `ATA.SeasAttr` function and its output.
- ***find.period*** **:** Find seasonal period(s) automatically. If NULL, 0 is default. When *find.period*,

  - ***0*** **:** none.
  - ***1*** **:** single period with `find.freq`.
  - ***2*** **:** single period with `forecast::findfrequency`.
  - ***3*** **:** multiple period with `find.freq` & stR.
  - ***4*** **:** multiple period with `find.freq` & tbats.

Accuracy Parameters **:**

- ***accuracy.type*** **:** Accuracy measure for selection of the best model. IF NULL, *sMAPE* is default.
  1. ***lik*** **:** maximum likelihood functions.
  2. ***sigma*** **:** residual variance.
  3. ***MAE*** **:** mean absolute error.
  4. ***MSE*** **:** mean square error.
  5. ***AMSE*** **:** average MSE over first 'nmse' forecast horizons.
  6. ***RMSE*** **:** root mean squared error.
  7. ***MPE*** **:** mean percentage error.
  8. ***MAPE*** **:** mean absolute percentage error.
  9. ***sMAPE*** **:** symmetric mean absolute percentage error.
  10. ***MASE*** **:** mean absolute scaled error.
  11. ***OWA*** **:** overall weighted average of MASE and sMAPE.
  12. ***MdAE*** **:** median absolute error.
  13. ***MdSE*** **:** median square error.
  14. ***RMdSE*** **:** root median squared error.
  15. ***MdPE*** **:** median percentage error.
  16. ***MdAPE*** **:** median absolute percentage error.
  17. ***sMdAPE*** **:** symmetric median absolute percentage error.
- ***nmse*** **:** If accuracy.type == "AMSE", "nmse" provides the number of steps for average multistep MSE ('2<=nmse<=30').

Transform Parameters **:**

- ***transform.order*** **:** If "before", Box–Cox transformation family will be applied, and then seasonal decomposition techniques will be applied. If "after", seasonal decomposition techniques will be applied, and then the Box–Cox transformation family will be applied.
- ***transform.method*** **:** Transformation methods: *Box_Cox, Sqrt, Reciprocal, Log, NegLog, Modulus, BickelDoksum, Manly, Dual, YeoJohnson, GPower, GLog* are used. Suppose the transformation process needs a shift parameter, `ATA.Transform` will calculate required the shift parameter automatically. When all types of Box–Cox family power techniques (except sqrt, reciprocal) are specified, *model.type* and *seasonal.type* is set to "A".
- ***transform.attr*** **:** Attributes set for Box–Cox transformation. If NULL, bcMethod = "loglik", bcLower = 0, bcUpper = 1, bcBiasAdj = FALSE. If you want to change, please use `ATA.BoxCoxAttr` function and its output.
- ***lambda*** **:** Box–Cox power transformation family parameter. If NULL, data transformed before the model is estimated.
- ***shift*** **:** Box–Cox power transformation family shifting parameter. If NULL, data transformed before the model is estimated. When *lambda* is specified, *model.type* and *seasonal.type* is set to "A".

Holdout Parameters **:**

- ***holdout*** **:** Default is FALSE. If TRUE, Ata method uses the holdout forecasting for accuracy measure to select the best model. In holdout forecasting, the last few data points are removed from the data series. The remaining historical data series is called in-sample data (training set), and the holdout data is called out-of-sample data (holdout set). If TRUE, holdout.set_size will used for holdout data.
- ***holdout.adjustedP*** **:** Default is TRUE. If TRUE, parP will be adjusted by the length of training-validation sets, and in-sample set when the holdout forecasting is active.
- ***holdout.set_size*** **:** If *holdout* is TRUE, this parameter will be the same as *h* for defining holdout set.
- ***holdin*** **:** Default is FALSE. If TRUE, Ata method uses the hold-in forecasting for accuracy measure to select the best model. In hold-in forecasting, the last h-length data points are used for accuracy measure.

Output of `ATA` function

Returns an object of class "ata", containing the generic access or functions `ATA.Forecast`, and `ATA.Accuracy` extracts the useful features of the value returned by "ata" and associated functions.

- ***actual*** **:** The original time series.
- ***fitted*** **:** Fitted values (one-step forecasts). The mean is of the fitted values is calculated over the ensemble.
- ***level*** **:** Estimated level values.
- ***trend*** **:** Estimated trend values.
- ***residuals*** **:** Original values minus fitted values.
- ***coefp*** **:** The weights attached to level observations.
- ***coefq*** **:** The weights attached to trend observations.
- ***p*** **:** Optimum level parameter.
- ***q*** **:** Optimum trend parameter.
- ***phi*** **:** Optimum damped trend parameter.
- ***model.type*** **:** Form of trend.
- ***h*** **:** The number of steps to forecast ahead.
- ***forecast*** **:** Point forecasts as a time series.
- ***out.sample*** **:** Test sets as a time series.
- ***method*** **:** The name of the optimum forecasting method as a character string.
- ***initial.level*** **:** Selected initial level values for the time series forecasting method.
- ***initial.trend*** **:** Selected initial trend values for the time series forecasting method.
- ***trend.opt*** **:** A choice of optional trend and level optimized trended methods (none, trend.fixed, or trend.search).
- ***transform.method*** **:** Box–Cox power transformation family methods are Box_Cox, Sqrt, Reciprocal, Log, NegLog, Modulus, BickelDoksum, Manly, Dual, YeoJohnson, GPower, GLog.
- ***transform.order*** **:** Define how to apply the Box–Cox power transformation techniques before or after seasonal decomposition.
- ***lambda*** **:** The Box–Cox power transformation family parameter.
- ***shift*** **:** The Box–Cox power transformation family shifting parameter.
- ***accuracy.type*** **:** Accuracy measure that is chosen for model selection.
- ***nmse*** **:** The number of steps for average multi-step MSE.
- ***accuracy*** **:** In-and out-sample accuracy measures and its descriptive that are calculated for optimum model are given.
- ***par.specs*** **:** Parameter sets for Information Criteria.
- ***holdout*** **:** Holdout forecasting is TRUE or FALSE.
- ***holdout.training*** **:** Training set in holdout forecasting.
- ***holdout.validation*** **:** Validation set in holdout forecasting.
- ***holdout.forecast*** **:** Holdout forecast.

- ***holdout.accuracy* :** Accuracy measure chosen for model selection in holdout forecasting.
- ***is.season* :** Indicates whether it contains seasonal pattern.
- ***seasonal.model* :** The name of the selected decomposition method.
- ***seasonal.type* :** Form of seasonality.
- ***seasonal.period* :** The number of seasonality periods (which defaults to `frequency(X)`).
- ***seasonal.index* :** Weights of seasonality.
- ***seasonal* :** Estimated seasonal values.
- ***seasonal.adjusted* :** Deseasonalized time series values.
- ***execution.time* :** The real and CPU time (in seconds) spent by the system executing that task, including the time spent executing run-time or system services on its behalf.
- ***calculation.time* :** How much real time (in seconds) the currently running R process has already taken.

Here are quick start examples using "aus_retail" dataset monthly retail turnover (in million AUD) in Australian states from April 1982 to December 2018 in the **tsibbledata** package.

```
library(tsibble)
library(tsibbledata)
library(lubridate)
library(dplyr)
library(tsbox)
library(ATAforecasting)

main_data <- aus_retail %>%
        filter(State == "New South Wales",
        Industry == "Department stores",
        `Series ID`== "A3349790V")
train_data <- tsbox::ts_ts(train_data)
test_data <- tail(train_data, 24)
train_data <- window(train_data, start = 1983, end = 2016.917)
ata_fit <- ATA(train_data, test_data, h=24)
ata_fit$is.season
unlist(ata_fit$accuracy$sMAPE)
unlist(ata_fit$accuracy$sMAPE$inSample)
unlist(ata_fit$accuracy$sMAPE$outSample)
unlist(ata_fit$accuracy$fits)
ata_fit$fitted
ata_fit$forecast
ata_fit$residuals
```

Here are some outputs for the above example from the **ATAforecasting** Package whose results are shown in Figure 9 and 10. 40 properties of the `ATA` module, including all results of the automatic forecasting using the Ata method are able to be obtained by using the "$" command as shown in the above example.

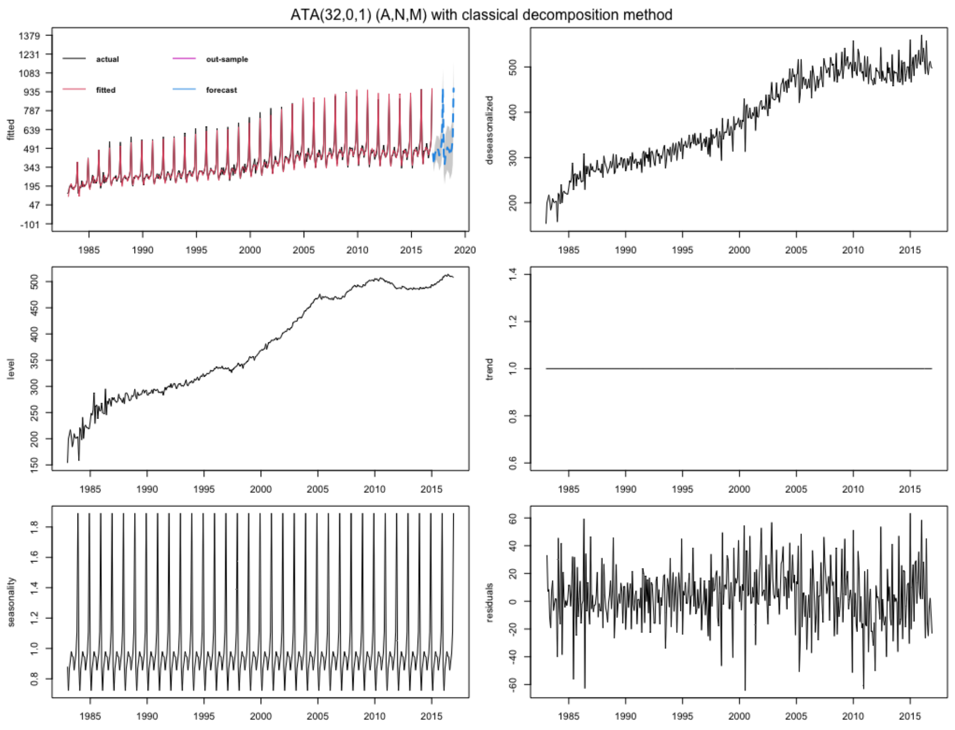
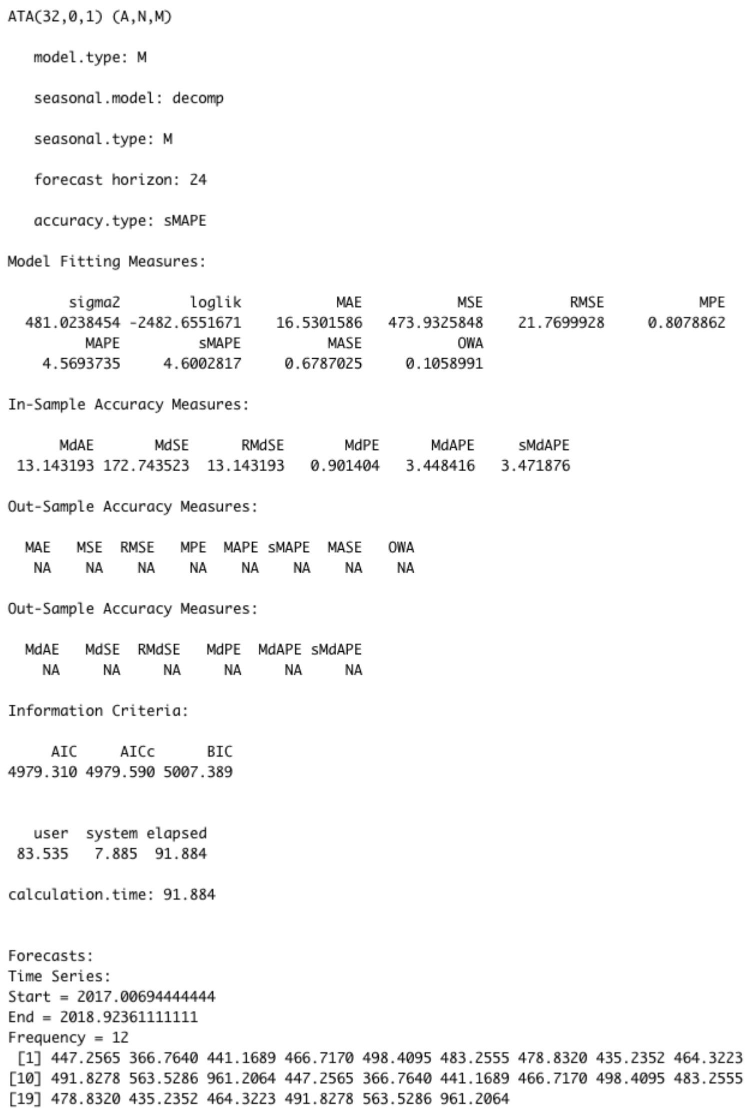


Figure 9: Forecasts from automatic ATA seasonal damped trended methods.

Another sample data is Makridakis Competitions 2000 (also known as the M-Competitions) monthly data in the **Mcomp** package (Hyndman et al., 2018).

```
atafit <- ATA(M3[[1899]]$x, M3[[1899]]$xx, parQ = 1, parPHI = 1
  , model.type = "A", seasonal.type = "M", seasonal.test = TRUE
  , seasonal.model = "decomp", level.fixed = FALSE, transform.method = "Box_Cox"
  , negative.forecast = FALSE)
```

Here are some outputs for the above example from the **ATAforecasting** Package. The results are shown in Figure 11.



```
ATA(32,0,1) (A,N,M)

   model.type: M

   seasonal.model: decomp

   seasonal.type: M

   forecast horizon: 24

   accuracy.type: sMAPE

Model Fitting Measures:

      sigma2        loglik           MAE           MSE          RMSE           MPE
  481.0238454 -2482.6551671    16.5301586   473.9325848    21.7699928     0.8078862
         MAPE         sMAPE          MASE           OWA
    4.5693735     4.6002817     0.6787025     0.1058991

In-Sample Accuracy Measures:

      MdAE       MdSE      RMdSE       MdPE      MdAPE     sMdAPE
 13.143193 172.743523  13.143193   0.901404   3.448416   3.471876

Out-Sample Accuracy Measures:

  MAE   MSE  RMSE   MPE  MAPE sMAPE  MASE   OWA
   NA    NA    NA    NA    NA    NA    NA    NA

Out-Sample Accuracy Measures:

  MdAE   MdSE  RMdSE   MdPE  MdAPE sMdAPE
    NA     NA     NA     NA     NA     NA

Information Criteria:

     AIC     AICc      BIC
4979.310 4979.590 5007.389


   user  system elapsed
 83.535   7.885  91.884

calculation.time: 91.884


Forecasts:
Time Series:
Start = 2017.00694444444
End = 2018.92361111111
Frequency = 12
 [1] 447.2565 366.7640 441.1689 466.7170 498.4095 483.2555 478.8320 435.2352 464.3223
[10] 491.8278 563.5286 961.2064 447.2565 366.7640 441.1689 466.7170 498.4095 483.2555
[19] 478.8320 435.2352 464.3223 491.8278 563.5286 961.2064
```

Figure 10: The default model output from the automatic ATA seasonal damped trended methods.

The object `atafit` is of class "ata" and contains all of the necessary information about the fitted model including model parameters, residuals, and so on. Printing the `atafit` object presents the main items of interest.

```
ATA.Forecast(atafit, h = 18, ci.level = 99
, negative.forecast = TRUE)
```

Some goodness-of-fit measures of forecast accuracy are obtained based on only the fitting data using `ATA.Accuracy`, we use the following commands.

```
ATA.Accuracy(atafit)
```

Fable modeling wrappers for ATAforecasting

We also developed a wrapper software (called **fable.ata** (Taylan et al., 2021b) to add the Ata method into the fable ecosystems using the **fabletools** (O'Hara-Wild et al., 2021b) package, which provides tools, helpers, and data structures for developing algorithms for the **fable** ecosystems (O'Hara-Wild et al., 2021a). Here are the quick start examples using the "aus_retail" dataset.

```
library(fable)
library(fable.ata)

fit <- aus_retail %>%
filter(State %in% c("New South Wales", "Victoria"),
Industry == "Department stores") %>%
```

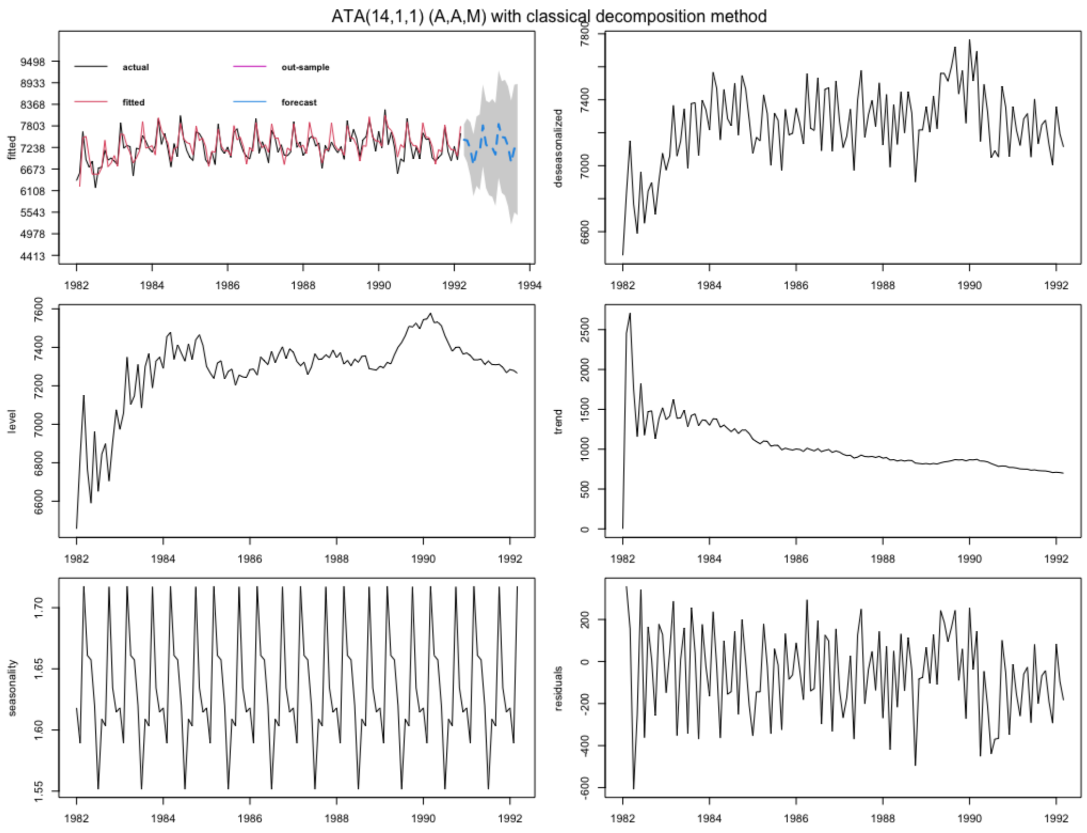


Figure 11: Forecasts from the automatic ATA seasonal trended methods for M3 sample.

```
model(
  ets = ETS(Turnover),
  arima = ARIMA(Turnover),
  snaive = SNAIVE(Turnover),
  ata = AutoATA(Turnover~trend("M") + season(type="M",method="stR"))
 ) %>%
mutate(mixed = (ets + arima + snaive + ata) / 4)
fc <- fit %>% forecast(h = 12)
fc %>% autoplot(filter(aus_retail, year(Month) > 2010), level = NULL)
```

Here are some outputs for the above example from the **fable** ecosystem functions (**fable** and **fable.ata** packages). The results are shown in Figure 12 and Figure 13.

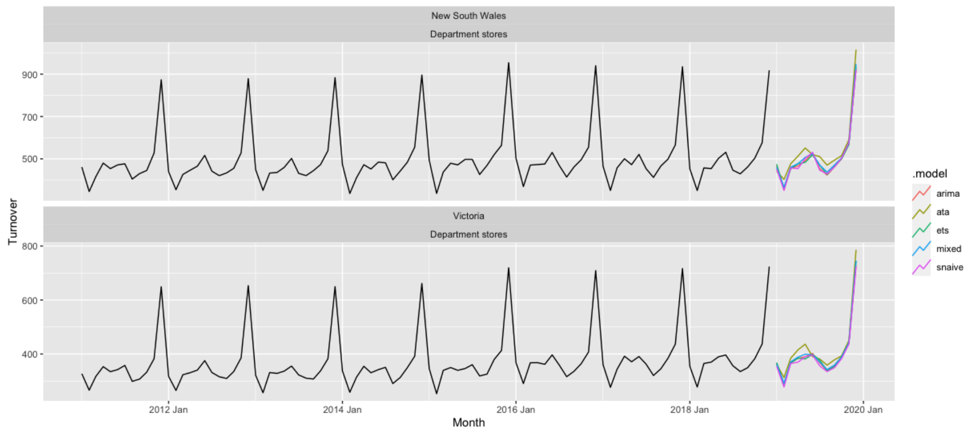
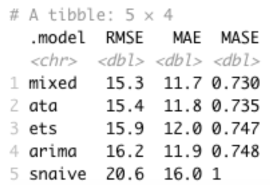


Figure 12: Forecasts from fable models for aus_retail dataset.

```
fit %>%
accuracy() %>%
group_by(.model) %>%
summarise(
        RMSE = mean(RMSE),
        MAE = mean(MAE),
        MASE = mean(MASE)
        ) %>%
arrange(MASE)
```



```
# A tibble: 5 × 4
  .model  RMSE   MAE  MASE
  <chr>  <dbl> <dbl> <dbl>
1 mixed   15.3  11.7 0.730
2 ata     15.4  11.8 0.735
3 ets     15.9  12.0 0.747
4 arima   16.2  11.9 0.748
5 snaive  20.6  16.0 1
```

Figure 13: Comparison of fable models accuracy measures.

Holdout forecasting

Using holdout samples is substantial implementation to fit a model where the epoch of fit is dissimilar to the epoch of assessment. According to this model evaluation procedure, the epoch of fit completes at any moment before the last observation, and the rest of the data are held out as a non-overlapping epoch of assessment. In concern with the epoch of fit, the holdout sample is an epoch in the future, used to compare the forecasting accuracy of model fits to past data.

The concept of a holdout sample is to split the in-sample data into two parts. The last few data points are taken out from the in-sample data. The leftover data is called the training set, and the removed data is called the validation set or holdout set. Assume k periods have been taken out as holdout samples from a total of T periods. The parameters are optimized by minimizing the fit accuracy measure for the first part of the data. After the parameters are optimized, for each model, computed multi-step forecasts over the period covered by the second part, or holdout sample. The models are then evaluated, comparing accuracy measures for these out-of-sample multi-step predictions of the holdout sample. The model whose out-of-sample predictions best fit the holdout sample is chosen. The selected model is refitted using all the data to get the final forecasting model.

The **ATAforecasting** package makes it easy to use the holdout sample method of model selection. The time range used to fit models and the time range used for model evaluation are able to independently controlled. To use holdout samples, the period of evaluation range to that last part of the data, and the period of fit range to the remainder of the data are set. The automatic model selection feature is able to be used to select the model whose multiperiod out-of-sample predictions best fit the holdout sample.

Now, a quick start example of how to call the holdout method in the package.

```
ata_holdout <- ATA(train_data, test_data, h=24, holdout = TRUE
  , holdout.set_size = 24, holdout.adjustedP = TRUE
  , seasonal.test = TRUE, seasonal.model = "decomp")
```

Applications

Ata method was proposed as an alternative to ES, and it is not a special case of it. The details on the method and how it helps solve some issues that ES suffers from can be found in Yapar et al. (2019), Yapar (2018), Yapar et al. (2018), Yapar et al. (2017). ATA can be adapted to all types of time series data and will always outperform its counter ES models.

There are many studies on the numerical and theoretical comparison of Box-Jenkins and ES methods. Several empirical studies have been published in turn by Reid (1969), Newbold and Granger (1974), Makridaki and Hibon (1979), Makridakis et al. (1982), Makridakis et al. (1993), Makridakis and Hibon (2000), Makridakis et al. (2018).

Efforts for better forecasting and the competitions in which the outcomes of these efforts are tested and measured will never cease. Better forecasting is crucial to every science and business field. The most important platforms in which the performance of the studies for accurate forecasting is measured are the M-Competitions (Hyndman, 2020). The most recent of these competitions, M4, has ended (Makridakis et al., 2018). The aim of the M4-Competition, like the competitions held before it, was to learn how to improve the forecasting accuracy, and how such learning can be applied to advance the theory and practice of forecasting, and are there any new methods that could really make a difference.

M-Competitions are very important and prestigious platforms for forecasting researchers since they provide researchers and developers of new forecasting methods opportunities to test and prove themselves. Another benefit of these competitions is that they usually lead to both the destruction of many taboos known in the forecasting literature and the discovery of new methods that help increase forecasting accuracy. The M-Competition was established by Spyros G. Makridakis in 1982 in a paper that studied the post-sample accuracy of several time series forecasting methods (Makridakis et al., 1982). The number of series was increased to 1001, and the data were subdivided into various categories (micro, macro, industry, demography, finance, other). The participants tested the accuracy of 24 methods on 1001 series with various horizons which were six for yearly data, eight for quarterly data, and eighteen for monthly data. The competition's goal was to explore how different procedures differ from each other and how information can be ensured that forecasters can make convenient choices under various conditions (Makridakis et al., 1984).

In Makridakis and Hibon (2000), the M3-Competition reports the reasons for conducting the competition and summarizes its outcomes. In the M3-Competition, 3003 series, composed of 6 different types of series and 4 different time intervals between successive observations. The three prior competitions have played a very major role in the forecasting literature. Their results ensured a basis for future forecasting research. Consequently, Makridakis et al. initiated the fourth competition. As per the Makridakis' team, the goal of the M4-Competition is to further study the utility and accuracy of various forecasting methods. Thus, the categories and number of the series and the forecasting methods are increased.

The M4-Competition is the progression of three previous competitions that began more than 45 years ago, whose objective was to learn how to evolve forecasting accuracy and how such learning can be implemented to proceed with the theory and performance of forecasting.

The purpose of M4 was to replicate the consequences of the prior ones and expand them into three aspects:

1. Substantially enhanced the number of series,
2. Contained machine learning forecasting methods,
3. Interpret both point forecasts and prediction intervals.

The some substantial outcomes of the M4-Competition are:

1. 12 of the 17 most accurate methods were "combinations" of mostly statistical approaches.
2. "hybrid" approach was a significant finding that use both statistical and machine learning features.

In the M4-Competition, the number of data from the previous M3-Competition (Makridakis and Hibon, 2000) was increased from 3,000 to 100,000. There were numerous applications (248), but only 49 of the applicants were able to provide forecasts for the entire 100,000 series. With the addition of 10 benchmarks and 2 standard methods, 61 methods were considered (Makridakis et al., 2020). Only 17 out of 49 valid applications outperformed the benchmark set by the competition committee. Of these 17 successful methods, 12 are combinations of known statistical methods obtained by using different weighting techniques.

The M3-Competition data set consists of 645 yearly, 756 quarterly, 1428 monthly, and 174 other series. The M4-Competition data set consists of 23000 yearly, 24000 quarterly, 48000 monthly, 359 weekly, 4227 daily, and 414 hourly series. The original data sets, as well as the forecasts of the methods that participated in the competitions, are available in the R packages **Mcomp** (Hyndman et al., 2018) and **M4comp2018** (BenTaieb, 2018).

In order to test and to apply this approach's forecasting performance on real data and compare it to the benchmarks and especially counter ES models, forecasts obtained from five versions of it given the shortcode numbers *Model-M3* and *Model-M4* are fitted to the M3 and M4 competitions.

Therefore, in this implementation, predetermined model parameters as defined in the following list items are used to obtain accurate forecasts. Results from seven different applications of the Ata method will be considered here.

1. $ATA(p, 0, 1)$ is an alternative to SES method where p is the optimum value for $q = 0$ with fixed damped trend ($\phi = 1$), and is where a simple model selection of the two models in $ATA^{add}(p, 0, 1)$ and $ATA^{mult}(p, 0, 1)$ is carried out based on selected in-sample accuracy measure.

```
# --- Code for Makridakis Competition 2018
# --- (M4 Forecasting Competition) Dataset.

# Load packages for creating plots
library(ATAforecasting)
library(M4comp2018)

fit1 <- ATA(M4[[1]]$x, M4[[1]]$xx, h = M4[[1]]$h, parQ = 0
, parPHI = 1 , seasonal.test = TRUE
, seasonal.model = "decomp", accuracy.type = "sMAPE"
, negative.forecast = FALSE)
```

2. $ATA^{add}(p, 1, 1)$ where p is optimized for $q = 1$ with fixed damped trend ($\phi = 1$)

```
fit2 <- ATA(M4[[1]]$x, M4[[1]]$xx, h = M4[[1]]$h, parQ = 1
, parPHI = 1, seasonal.test = TRUE, seasonal.model = "decomp"
, model.type = "A", accuracy.type = "sMAPE"
, negative.forecast = FALSE)
```

3. $ATA - comb$ where a simple average of the forecasts from the two models in (1) and (2) is used as a forecast.

```
fit3 <- (fit1 + fit2) / 2
```

4. $ATA^{add}(p, 1, \phi)$ is an alternative to damped trend method where q is optimized for $p = p^*$ with damped trend.

```
fit4 <- ATA(M4[[1]]$x, M4[[1]]$xx, h = M4[[1]]$h, parQ = 1
, start.phi = 0.80, end.phi = 1, size.phi = 0.01
, seasonal.test = TRUE, seasonal.model = "decomp"
, model.type = "A", accuracy.type = "sMAPE"
, negative.forecast = FALSE)
```

Model encoded by Model-M4 fits the $ATA^{add}(p, 1, \phi)$ to the yearly data sets and uses the $ATA - comb$, a simple average of the forecasts obtained from the models $ATA^{add}(p, 1, 1)$ and $ATA(p, 0, 1)$, for the other data sets in M4-Competitions. Model-M3 fits $ATA^{add}(p, 0, 1)$, $ATA(p, 0, 1)$, and uses the $ATA - comb$ for the data sets in M3-Competitions.

Team	Method Type	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	Total	Rank
Smyl	Hybrid	13.176	9.679	12.126	7.817	3.170	9.328	11.374	1
Montero-Manso, et al.	Combination (S & ML)	13.528	9.733	12.639	7.625	3.097	11.506	11.720	3
Pawlikowski, et al.	Combination (S)	13.943	9.796	12.747	6.919	2.452	9.611	11.845	5
Jaganathan. & Prakash	Combination (S & ML)	13.712	9.809	12.487	6.814	3.037	9.934	11.695	2
Fiorucci & Louzada	Combination (S)	13.673	9.816	12.737	8.627	2.985	15.563	11.836	4
Petropoulos & Svetunkov	Combination (S)	13.669	9.800	12.888	6.726	2.995	13.167	11.887	6
Shaub	Combination (S)	13.679	10.378	12.839	7.818	3.222	13.466	12.020	9
Legaki & Koutsouri	Statistical	13.366	10.155	13.002	9.148	3.041	17.567	11.986	8
Doornik, et al.	Combination (S)	13.910	10.000	12.780	6.728	3.053	8.913	11.924	7
Pedregal, et al.	Combination (S)	13.821	10.093	13.151	8.989	3.026	9.765	12.114	13
Model-M4	Statistical	13.930	10.292	12.936	8.540	3.095	12.851	12.098	11
Spiliotis & Assimakopoulos	Statistical	13.804	10.128	13.142	8.990	3.027	17.756	12.148	15
Roubinchtein	Combination (S)	14.445	10.172	12.911	8.435	3.270	12.871	12.183	17
Ibrahim	Statistical	13.677	10.089	13.321	9.089	3.071	18.093	12.198	18
Tartu M4 seminar	Combination (S & ML)	14.096	11.109	13.290	8.513	2.852	13.851	12.496	23
Waheeb	Combination (S)	14.783	10.059	12.770	7.076	2.997	12.047	12.146	14
Darin & Stellwagen	Statistical	14.663	10.155	13.058	6.582	3.077	11.683	12.279	19
Dantas & Cyrino Oliveira	Combination (S)	14.746	10.254	13.462	8.873	3.245	16.941	12.553	25
The M4 Team (Theta)	Statistical	14.593	10.311	13.002	9.093	3.053	18.138	12.309	20
The M4 Team (Com)	Statistical	14.848	10.175	13.434	8.944	2.980	22.053	12.555	27
The M4 Team (Arima)	Statistical	15.168	10.431	13.443	8.653	3.193	12.045	12.661	29
The M4 Team (Damped)	Statistical	15.198	10.237	13.473	8.866	3.064	19.265	12.661	30
The M4 Team (ETS)	Statistical	15.356	10.291	13.525	8.727	3.046	17.307	12.725	31
The M4 Team (Holt)	Statistical	16.354	10.907	14.812	9.708	3.066	29.249	13.775	43
The M4 Team (SES)	Statistical	16.396	10.600	13.618	9.012	3.045	18.094	13.087	37

Table 2: Average forecasting errors for various data types and overall ranks with respect to sMAPE.

The forecasting performance of the Model-M4 that competed in the M4-Competition are given in the following three tables (Tables 2, 3, and 4) with respect to the error criteria sMAPE, MASE, and OWA, respectively.

According to sMAPE (Table 2), the *Model-M4* of the ATA models is ranked in the first 20. The *Model-M4* performs much better than ETS despite the fact that only sMAPE was used for optimizing the ATA approaches for the in-sample data, and these approaches only considered limited numbers of candidate models to choose from, unlike ETS.

Team	Method Type	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	Total	Rank
Smyl	Hybrid	2.980	1.118	0.884	2.356	3.446	0.893	1.536	1
Montero-Manso, et al.	Combination (S & ML)	3.060	1.111	0.893	2.108	3.344	0.819	1.551	3
Pawlikowski, et al.	Combination (S)	3.130	1.125	0.905	2.158	2.642	0.873	1.547	2
Jaganathan. & Prakash	Combination (S & ML)	3.126	1.135	0.895	2.350	3.258	0.976	1.571	6
Fiorucci & Louzada	Combination (S)	3.046	1.122	0.907	2.368	3.194	1.203	1.554	4
Petropoulos & Svetunkov	Combination (S)	3.082	1.118	0.913	2.133	3.229	1.458	1.565	5
Shaub	Combination (S)	3.038	1.198	0.929	2.947	3.479	1.372	1.595	7
Legaki & Koutsouri	Statistical	3.009	1.198	0.966	2.601	3.254	2.557	1.601	8
Doornik, et al.	Combination (S)	3.262	1.163	0.931	2.302	3.284	0.801	1.627	11
Pedregal, et al.	Combination (S)	3.185	1.164	0.943	2.488	3.232	1.049	1.614	10
Model-M4	Statistical	3.117	1.231	0.962	2.578	3.277	2.238	1.631	13
Spiliotis & Assimakopoulos	Statistical	3.184	1.178	0.959	2.488	3.232	1.808	1.628	12
Roubinchtein	Combination (S)	3.244	1.159	0.921	2.290	3.632	1.129	1.633	15
Ibrahim	Statistical	3.075	1.185	0.977	2.583	3.894	2.388	1.644	16
Tartu M4 seminar	Combination (S & ML)	3.091	1.250	1.002	2.375	3.025	1.058	1.633	14
Waheeb	Combination (S)	3.400	1.160	1.029	2.180	3.321	0.861	1.706	27
Darin & Stellwagen	Statistical	3.406	1.168	0.924	2.107	4.128	0.856	1.693	25
Dantas & Cyrino Oliveira	Combination (S)	3.294	1.170	0.952	2.534	3.436	1.598	1.657	17
The M4 Team (Theta)	Statistical	3.382	1.232	0.970	2.637	3.262	2.455	1.696	26
The M4 Team (Com)	Statistical	3.280	1.173	0.966	2.432	3.203	4.582	1.663	18
The M4 Team (Arima)	Statistical	3.402	1.165	0.930	2.556	3.410	0.943	1.666	19
The M4 Team (Damped)	Statistical	3.379	1.173	0.972	2.404	3.236	2.956	1.683	23
The M4 Team (ETS)	Statistical	3.444	1.161	0.948	2.527	3.253	1.824	1.680	21
The M4 Team (Holt)	Statistical	3.550	1.198	1.009	2.420	3.223	9.356	1.772	34
The M4 Team (SES)	Statistical	3.981	1.340	1.019	2.685	3.281	2.385	1.885	39

Table 3: Average forecasting errors for various data types and overall ranks with respect to MASE.

Team	Method Type	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	Total	Rank
Smyl	Hybrid	0.778	0.847	0.836	0.851	1.046	0.440	0.821	1
Montero-Manso, et al.	Combination (S & ML)	0.799	0.847	0.858	0.796	1.019	0.484	0.838	2
Pawlikowski, et al.	Combination (S)	0.820	0.855	0.867	0.766	0.806	0.444	0.841	3
Jaganathan & Prakash	Combination (S & ML)	0.813	0.859	0.854	0.795	0.996	0.474	0.842	4
Fiorucci & Louzada	Combination (S)	0.802	0.855	0.868	0.897	0.977	0.674	0.843	5
Petropoulos & Svetunkov	Combination (S)	0.806	0.853	0.876	0.751	0.984	0.663	0.848	6
Shaub	Combination (S)	0.801	0.908	0.882	0.957	1.060	0.653	0.860	7
Legaki & Koutsouri	Statistical	0.788	0.898	0.905	0.968	0.996	1.012	0.861	8
Doornik, et al.	Combination (S)	0.836	0.878	0.881	0.782	1.002	0.410	0.865	9
Pedregal, et al.	Combination (S)	0.824	0.883	0.899	0.939	0.990	0.485	0.869	11
Model-M4	Statistical	0.818	0.916	0.901	0.930	1.008	0.817	0.872	12
Spiliotis & Assimakopoulos	Statistical	0.823	0.889	0.907	0.939	0.990	0.860	0.874	13
Roubinchtein	Combination (S)	0.850	0.885	0.881	0.873	1.091	0.586	0.876	14
Ibrahim	Statistical	0.805	0.890	0.921	0.961	1.098	0.991	0.880	15
Tartu M4 seminar	Combination (S & ML)	0.820	0.960	0.932	0.892	0.930	0.598	0.888	17
Waheeb	Combination (S)	0.880	0.880	0.927	0.779	0.999	0.507	0.894	18
Darin & Stellwagen	Statistical	0.877	0.887	0.887	0.739	1.135	0.496	0.895	19
Dantas & Cyrino Oliveira	Combination (S)	0.866	0.892	0.914	0.941	1.057	0.794	0.896	20
The M4 Team (Theta)	Statistical	0.872	0.917	0.907	0.971	0.999	1.006	0.897	21
The M4 Team (Com)	Statistical	0.867	0.890	0.920	0.926	0.978	1.556	0.898	22
The M4 Team (Arima)	Statistical	0.892	0.898	0.903	0.932	1.044	0.524	0.902	23
The M4 Team (Damped)	Statistical	0.890	0.893	0.924	0.917	0.997	1.141	0.907	25
The M4 Team (ETS)	Statistical	0.903	0.891	0.915	0.931	0.996	0.852	0.908	26
The M4 Team (Holt)	Statistical	0.947	0.932	0.988	0.966	0.995	2.749	0.971	37
The M4 Team (SES)	Statistical	1.003	0.970	0.951	0.975	1.000	0.990	0.975	39

Table 4: Average forecasting errors for various data types and overall ranks with respect to OWA.

The forecasting performance of the Model-M3 that competed in the M4-Competition are given in the Table 5 with the error criteria sMAPE.

These results should motivate users to consider ATA instead of ES-based forecasting. An important result from the M4-Competition was that combining forecasts improved accuracy. This improvement will become even stronger if the set of initial candidate models are chosen wisely and more meaningful if the combination can be obtained faster. Speed is an undeniable factor when choosing a forecasting method due to the need to obtain forecasts for the streaming and big data sets. The results obtained by using a simple combination of ARIMA and ATA for the M4-Competition data set are given in Table 6. For all error metrics considered, ATA approaches provide much better forecasts, and since the optimization is much faster than ETS, these more satisfying forecasts are obtained much faster.

Just by using the simple combination of ATA and ARIMA, forecasts that are more accurate than most of the methods that competed in the M4-Competition and that can compete with the more

	Forecasting horizons										Averages					
Method	1	2	3	4	5	6	8	12	15	18	1-4	1-6	1-8	1-12	1-15	1-18
Naive2	10.5	11.3	13.6	15.1	15.1	15.9	14.5	16.0	19.3	20.7	12.62	13.57	13.76	14.24	14.81	15.47
Single	9.5	10.6	12.7	14.1	14.3	15.0	13.3	14.5	18.3	19.4	11.73	12.71	12.84	13.13	13.67	14.32
Holt	9.0	10.4	12.8	14.5	15.1	15.8	13.9	14.8	18.8	20.2	11.67	12.93	13.11	13.42	13.95	14.60
Winter	9.1	10.5	12.9	14.6	15.1	15.9	14.0	14.6	18.9	20.2	11.77	13.01	13.19	13.48	14.01	14.65
Dampen	8.8	10.0	12.0	13.5	13.8	14.3	12.5	13.9	17.5	18.9	11.07	12.05	12.17	12.45	12.98	13.64
Comb (S-H-D)	8.9	10.0	12.0	13.5	13.7	14.2	12.4	13.6	17.3	18.3	11.10	12.04	12.13	12.4	12.91	13.52
ETS	8.8	9.8	12.0	13.5	13.9	14.7	13.0	14.1	17.6	18.9	11.04	12.13	12.32	12.66	13.14	13.77
$ATA(p,0,1)$	8.9	10.0	12.1	13.7	13.9	14.7	12.8	13.9	17.3	18.9	11.16	12.21	12.34	12.64	13.13	13.77
$ATA(p,1,1)$	8.4	9.7	11.5	12.9	13.6	14.2	12.9	15.4	18.9	20.9	10.64	11.72	11.94	12.66	13.32	14.09
$ATA(p,q,\phi = 0.5)$	8.6	9.6	11.6	13.2	13.5	14.2	12.4	13.7	17.0	18.6	10.76	11.77	11.92	12.24	12.75	13.39
Model-M3	8.5	9.6	11.4	12.8	13.0	13.6	12.0	13.1	16.3	17.4	10.56	11.47	11.58	11.94	12.40	12.94

Table 5: Average sMAPE across different forecast horizons: all 3003 series.

	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	Total
	sMAPE						
ETS & ARIMA	14.691	10.027	12.917	8.439	3.076	14.377	12.205
Model-M3 & ARIMA	13.847	9.987	12.653	7.607	2.998	11.942	11.859
	MASE						
ETS & ARIMA	3.334	1.132	0.909	2.476	3.259	1.249	1.627
Model-M3 & ARIMA	3.093	1.148	0.908	2.345	3.255	1.436	1.575
	OWA						
ETS & ARIMA	0.869	0.868	0.875	0.906	1.002	0.652	0.875
Model-M3 & ARIMA	0.813	0.872	0.866	0.837	0.989	0.625	0.849

Table 6: Average forecasting errors for various data types and error metrics using simple combinations of forecasts.

accurate methods considering the computation complexity and time as important factors can be obtained. The results are given along with the ranks when all the methods are ranked according to OWA in Table 7. The three simple combinations of ATA and ARIMA are ranked in the top 10 when all other methods are considered.

Team	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	Total	Rank
Smyl	0.778	0.847	0.836	0.851	1.046	0.440	0.821	1
Montero-Manso, et al.	0.799	0.847	0.858	0.796	1.019	0.484	0.838	2
Pawlikowski, et al.	0.820	0.855	0.867	0.766	0.806	0.444	0.841	3
Jaganathan. & Prakash	0.813	0.859	0.854	0.795	0.996	0.474	0.842	4
Fiorucci & Louzada	0.802	0.855	0.868	0.897	0.977	0.674	0.843	5
Petropoulos & Svetunkov	0.806	0.853	0.876	0.751	0.984	0.663	0.848	6
Model-M4 & ARIMA	0.813	0.872	0.866	0.837	0.989	0.625	0.849	8
Shaub	0.801	0.908	0.882	0.957	1.060	0.653	0.860	10
Legaki & Koutsouri	0.788	0.898	0.905	0.968	0.996	1.012	0.861	11
Doornik, et al.	0.836	0.878	0.881	0.782	1.002	0.410	0.865	12

Table 7: Average forecasting errors (OWA) for various data types along with the ranks.

Conclusion

In this study, we have introduced a novel method of bagging for the Ata method using power family transformations and various seasonal decomposition techniques. Ata method is a new and simple forecasting method that is an alternative to exponential smoothing. Although the Ata method's form is analogous to exponential smoothing, its weighting and parameterization schemes are utterly particular. Therefore, it is not a specific case of ES. It can be adapted to all types of time series data, much like ES and ARIMA, in addition to providing more accurate forecasts. Also, ATA can be optimized faster than exponential smoothing since its parameters can take on a limited number of discrete values only.

The goal of this manuscript is to introduce a new package for a new univariate time series forecasting method that provides innovative solutions to issues faced during the initialization and optimization stages of existing methods. The **ATAforecasting** package implements several different routines, most of which are related to the Ata method. Nevertheless, its modular structure enables the user to customize and complement the included functionality by means of custom algorithms or even other R packages. The **ATAforecasting** package provides a more general-purpose development as a comprehensive toolkit for automatic time series forecasting without any expertise on the R program. It focuses on modeling all types of time series components with any preferred Ata method and handling seasonality patterns by utilizing some popular decomposition techniques. Also, it combines several stationarity and seasonality tests, Box–Cox transformations, seasonal decomposition techniques with the Ata method. **ATAforecasting** performance is superior to existing methods both in terms of easy implementation, accurate, and flexible forecasting framework.

The **ATAforecasting** package categorizes some of the best-known techniques into three groups: (a) power transformation-based methods, (b) decomposition-based methods, and (c) time series forecasting-based methods. The package is also designed to assist research along with the whole modeling process: data preparation, model selection, prediction and forecasting, and interpretation of outcomes handling summaries and demonstrating functionalities. Providing these combinations of methods to the users is considered to introduce a new decomposition-based approach to time series forecasting with the Ata method, to provide automation, optimization, and bagging of the Ata method, which is an innovative and accurate univariate time series analysis method without any expertise by R program. Specifically, a proposed analytical methodology of the time series method with the**ATAforecasting** R package combines several stationarity and seasonality tests, power family transformations, and various seasonal decomposition techniques with the Ata method. In addition to this theoretical model, we focus on the computational implementation of all considered Ata methods in the **ATAforecasting** package. In particular, simulation and estimation have been demonstrated. Besides, the **ATAforecasting** package is aligned to many worthy R packages, such as **forecast**, **urca**, **uroot**, **seasonal**, **stR**, **stlplus**, **xts**, **timeSeries**, **TSA**, **tseries**.

In the future, the package should be extended to provide a comprehensive set of tools for three common issues in forecast combination prior to estimation, fast optimization of model parameters, missing values, and modeling with regressor variables. Users would have the option to automate the selection algorithm so that a good combination method is found based on the training set fit. Finally, the package offers specialized functions for summarizing and visualizing the combination results. Along this vein, a class for model specifications should be added alongside the actual implementations via arguments for the fitting functions. In that way, the package can be aligned to M-Competition benchmark time series models and useful R package. Furthermore, the package could benefit from robust estimation methods, another focus for future research.

Acknowledgement

This article was produced from Ali Sabri Taylan's PhD dissertation.

Bibliography

M. Adya, J. S. Armstrong, F. Collopy, and M. Kennedy. An application of rule-based forecasting to a situation lacking domain knowledge. *International Journal of Forecasting*, 16(4):477–484, 2000. [p508]

J. S. Armstrong. Combining forecasts: The end of the beginning or the beginning of the end? *International Journal of Forecasting*, 5(4):585–588, 1989. [p508]

J. S. Armstrong. *Principles of forecasting: a handbook for researchers and practitioners*, volume 30. Springer Science & Business Media, 2001. [p508]

V. Assimakopoulos and K. Nikolopoulos. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, 2000. [p508]

J. M. Bates and C. W. Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969. [p508]

S. BenTaieb. *M4comp: Data from the M4 Time Series Forecasting Competition*, 2018. URL `https://github.com/cran/M4comp`. R package version 0.0.1. [p523, 532]

C. Bergmeir, R. J. Hyndman, and J. M. Benítez. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International Journal of Forecasting*, 32(2):303–312, 2016. [p507]

P. J. Bickel and K. A. Doksum. An analysis of transformations revisited. *Journal of the American Statistical Association*, 76(374):296–311, 1982. [p511]

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964. [p510, 511]

R. G. Brown. *Statistical forecasting for inventory control*. McGraw/Hill, 1959. [p508]

J. P. Burman. Seasonal adjustment by signal extraction. *Journal of the Royal Statistical Society. Series A (General)*, 143(3):321–337, 1980. [p515]

F. Canova and B. E. Hansen. Are seasonal patterns constant over time? a test for seasonal stability. *Journal of Business and Economic Statistics*, 13(3):237–252, 1995. [p514]

R. J. Carroll. A robust method for testing transformations to achieve approximate normality. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(1):71–78, 1980. [p510]

B. Cetin and I. Yavuz. Comparison of forecast accuracy of Ata and exponential smoothing. *Journal of Applied Statistics*, 0(0):1–11, 2020. doi: 10.1080/02664763.2020.1803813. [p507]

K.-S. Chan and B. Ripley. *TSA: Time Series Analysis*, 2020. URL `https://cran.r-project.org/package=TSA`. R package version 1.3. [p523]

G. Chen, R. A. Lockhart, and M. A. Stephens. Box-cox transformations in linear models: Large sample theory and tests of normality. *Canadian Journal of Statistics*, 30(2):177–209, 2002. [p510]

R. T. Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989. [p508]

R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990. [p508, 515, 518, 524]

J. J. Commandeur, S. J. Koopman, and M. Ooms. Statistical software for state space methods. *Journal of Statistical Software*, 41(1), 2011. URL `https://www.jstatsoft.org`. [p515]

M. T. Copeland. Statistical indices of business conditions. *The Quarterly Journal of Economics*, 29(3): 522–562, 1915. [p515]

E. B. Dagum. *X-11-ARIMA/88 Seasonal Adjustment Method - Foundations and Users' Manual*. Statistics Canada, 1988. URL `https://www.census.gov/ts/papers/Emanual.pdf`. [p515]

E. B. Dagum and S. Bianconcini. *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation*. Springer, 2016. [p515]

R. Davidson and J. MacKinnon. *Estimation and Inference in Econometrics*. Oxford University Press, 1993. [p513]

J. L. de Lacalle. *uroot: Unit Root Tests for Seasonal Time Series*, 2020. URL `https://cran.r-project.org/package=uroot`. R package version 2.1-2. [p523]

D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979. [p513]

D. A. Dickey, D. P. Hasza, and W. A. Fuller. Testing for unit roots in seasonal time series. *Journal of the American Statistical Association*, 79(386):355–367, 1984. [p514]

A. Dokumentov and R. J. Hyndman. Str: A seasonal-trend decomposition procedure based on regression. Monash Econometrics and Business Statistics Working Papers 13/15, Monash University, Department of Econometrics and Business Statistics, 2015. URL `https://EconPapers.repec.org/RePEc:msh:ebswps:2015-13`. [p515, 520, 524]

A. Dokumentov and R. J. Hyndman. *stR: STR Decomposition*, 2018. URL `https://cran.r-project.org/package=stR`. R package version 0.4. [p509, 523]

B. P. Durbin, J. S. Hardin, D. M. Hawkins, and D. M. Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18(1):105–110, 2002. [p511]

D. Eddelbuettel, R. Francois, J. J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2020a. URL `https://cran.r-project.org/package=Rcpp`. R package version 1.0.5. [p523]

D. Eddelbuettel, R. Francois, D. Bates, and B. Ni. *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*, 2020b. URL https://cran.r-project.org/package=RcppArmadillo. R package version 0.10.1.2.0. [p523]

D. F. Findley. Some recent developments and directions in seasonal adjustment. *Journal of official statistics*, 21(2):343, 2005. [p515]

D. F. Findley, B. C. Monsell, W. R. Bell, M. C. Otto, and B.-C. Chen. New capabilities and methods of the x-12-arima seasonal-adjustment program. *Journal of Business & Economic Statistics*, 16(2):127–152, 1998. [p515]

E. S. Gardner and E. McKenzie. Forecasting trends in time series. *Management Science*, 31(10):1237–1246, 1985. [p508]

V. M. Guerrero. Time-series analysis supported by power transformations. *Journal of Forecasting*, 12(1): 37–48, 1993. [p511]

R. Hafen. *stlplus: Enhanced Seasonal Decomposition of Time Series by Loess*, 2016. URL https://cran.r-project.org/package=stlplus. R package version 0.5.1. [p509, 523]

R. P. Hafen. *Local regression models: Advancements, applications, and new methods*. PhD thesis, Purdue University, 2010. [p519]

J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994. [p513]

A. K. Han. A non-parametric analysis of transformations. *Journal of Econometrics*, 35(2–3):191–209, 1987. [p510]

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990. [p515]

F. Hayashi. *Econometrics*. Princeton University Press, 2000. [p513]

F. Hernandez and R. A. Johnson. The large-sample behavior of transformations to normality. *Journal of the American Statistical Association*, 75(372):855–861, 1980. [p510]

J. Horowitz. *Semiparametric and Nonparametric Methods in Econometrics*. Springer, 2009. [p510]

S. Hylleberg, R. F. Engle, C. W. J. Granger, and B. S. Yoo. Seasonal integration and cointegration. *Journal of Econometrics*, 1344(1):215–238, 1990. [p514]

R. J. Hyndman. A brief history of forecasting competitions. *International Journal of Forecasting*, 36(1): 7–14, 2020. [p532]

R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2019. URL https://otexts.com/fpp3/. [p508, 522]

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. [p522]

R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, 18(3):439–454, 2002. [p508]

R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer-Verlag, 2008. [p508]

R. J. Hyndman, M. Akram, C. Bergmeir, and M. O'Hara-Wild. *Mcomp: Data from the M-Competitions*, 2018. URL https://cran.r-project.org/package=Mcomp. R package version 2.8. [p523, 528, 532]

R. J. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen. *forecast: Forecasting functions for time series and linear models*, 2020. URL https://cran.r-project.org/package=forecast. R package version 8.13. [p508, 509, 523]

J. A. John and N. R. Draper. An alternative family of transformations. *Journal of the Royal Statistical Society Series C*, 29(2):190–197, 1980. [p511]

D. M. Kelmansky, E. J. Martinez, and V. Leiva. A new variance stabilizing transformation for gene expression data analysis. *Statistical Applications in Genetics and Molecular Biology*, 12(6):653–666, 2013. [p511]

A. J. Koning, P. H. Franses, M. Hibon, and H. O. Stekler. The m3 competition: Statistical tests of the results. *International Journal of Forecasting*, 21(3):397–409, 2005. [p508]

D. Kwiatkowski, P. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1–3):159–178, 1992. [p513]

A. M. D. Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011. [p508, 515, 519, 524]

G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2): 297–303, 1978. [p514]

F. R. Macaulay. *Appendix I - The Measurement of Probable Seasonal Fluctuations By Means of Operations on the Deviations of the Data*, pages 121–136. NBER, 1931. URL `http://www.nber.org/chapters/c9369`. [p515]

S. G. Makridaki and M. Hibon. Accuracy of forecasting: an empirical investigation. *Journal of Royal Statistical Society*, 142(2):97–145, 1979. [p531]

S. Makridakis and R. L. Winkler. Averages of forecasts: Some empirical results. *Management Science*, 29 (9):987–996, 1983. [p508]

S. G. Makridakis and M. Hibon. The m3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. [p508, 531, 532]

S. G. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982. [p508, 531, 532]

S. G. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. *The forecasting accuracy of major time series methods*. Wiley, 1984. [p508, 532]

S. G. Makridakis, M. Hibon, S. Chatfield, M. Lawrence, T. Mills, K. Ord, and L. F. Simmons. The m-2 competition: a real time judgementally based forecasting study. *International Journal of Forecasting*, 9 (2):5–23, 1993. [p531]

S. G. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018. [p508, 531, 532]

S. G. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. [p532]

S. Meintanis and G. Stupfler. Transformations to symmetry based on the probability weighted characteristic function. *KYBERNETIKA*, 51(4):571–587, 2015. [p510]

B. C. Monsell. The x-13a-s seasonal adjustment program. In *Proceedings of the 2007 Federal Committee On Statistical Methodology Research Conference. URL http://www. fcsm. gov/07papers/Monsell. II-B. pdf*, 2007. [p515]

B. C. Monsell, J. A. D. Aston, and S. J. Koopman. Toward x-13? Technical report, United States Census Bureau, 2003. URL `https://www.census.gov/content/dam/Census/library/working-papers/2003/adrm/jsm2003bcm.pdf`. [p515]

Y. Mu and X. He. Power transformation toward a linear regression quantile. *Journal of the American Statistical Association*, 102(477):269–279, 2007. [p510]

P. Newbold and C. W. Granger. Experience with forecasting univariate time series and the combination of forecast. *Journal of Royal Statistical Society*, 137(2):131–165, 1974. [p531]

M. O'Hara-Wild, R. Hyndman, E. Wang, G. Caceres, T.-G. Hensel, and T. Hyndman. *fable: Forecasting Models for Tidy Time Series*, 2021a. URL `https://cran.r-project.org/package=fable`. R package version 0.3.1. [p529]

M. O'Hara-Wild, R. Hyndman, E. Wang, D. Cook, G. Athanasopoulos, and D. Holt. *fabletools: Core Tools for Packages in the 'fable' Framework*, 2021b. URL `https://cran.r-project.org/package=fabletools`. R package version 0.3.1. [p529]

D. R. Osborn, A. P. L. Chui, J. Smith, and C. R. Birchenhall. Seasonality and the order of integration for consumption. *Oxford Bulletin of Economics and Statistics*, 50(4):361–377, 1988. [p514]

C. C. Pegels. On startup or learning curves: An expanded view. *AIIE Transactions*, 1(3):216–222, 1969. [p508]

W. M. Persons. *Indices of Business Conditions: An Index of General Business Conditions*, volume 1. Harvard University Press, 1919. [p515]

B. Pfaff, E. Zivot, and M. Stigler. *urca: Unit Root and Cointegration Tests for Time Series Data*, 2016. URL `https://cran.r-project.org/package=urca`. R package version 1.3-0. [p523]

P. C. B. Phillips and P. Perron. Testing for a unit root in time series regression. *Biometrika*, 75(2):335–346, 1988. [p513]

A. J. Quiroz, M. Nakamura, and F. J. Pérez. Estimation of a multivariate box-cox transformation to elliptical symmetry via the empirical characteristic function. *Annals of the Institute of Statistical Mathematics*, 48(4):687–709, 1996. [p510]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL `https://www.R-project.org/`. ISBN 3-900051-07-0. [p508, 509, 522]

D. J. Reid. A comparative study of time series prediction techniques on economic data. *International Journal of Forecasting*, 19(2):303–308, 1969. [p531]

J. A. Ryan, J. M. Ulrich, R. Bennett, and C. Joy. *xts: eXtensible Time Series*, 2020. URL `https://cran.r-project.org/package=xts`. R package version 0.12.1. [p523]

S. E. Said and D. A. Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607, 1984. [p513]

R. M. Sakia. The box-cox transformation technique: A review. *Journal of the Royal Statistical Society Series D*, 41(2):169–178, 1992. [p510]

C. Sax and D. Eddelbuettel. *seasonal: R Interface to X-13-ARIMA-SEATS*, 2020. URL `https://cran.r-project.org/package=seasonal`. R package version 1.7.1. [p523]

J. Shiskin. *Types of Economic Fluctuations*, pages 5–6. NBER, 1957. URL `http://www.nber.org/chapters/c2726`. [p515]

J. Shiskin, A. H. Young, and J. C. Musgrave. The x-11 variant of the census-ii method seasonal adjustment program. Technical Report 15, Bureau of the U.S. Census, 02 1967. URL `https://www.census.gov/ts/papers/ShiskinYoungMusgrave1967.pdf`. [p515]

J. G. Staniswalis, T. A. Severini, and P. G. Moschopoulos. On a data based power transformation for reducing skewness. *Journal of Statistical Computation and Simulation*, 46(1-2):91–100, 1993. [p510]

A. S. Taylan, H. T. Selamlar, and G. Yapar. *ATAforecasting: Automatic Time Series Analysis and Forecasting Using the Ata Method*, 2021a. URL `https://cran.r-project.org/package=ATAforecasting`. R package version 0.0.55. [p508]

A. S. Taylan, H. T. Selamlar, and G. Yapar. *fable.ata: 'ATAforecasting' Modelling Interface for 'fable'*, 2021b. URL `https://cran.r-project.org/package=fable.ata`. R package version 0.0.2. [p529]

A. Trapletti and K. Hornik. *tseries: Time Series Analysis and Computational Finance*, 2020. URL `https://CRAN.R-project.org/package=tseries`. R package version 0.10-48. [p523]

J. W. Tukey. *Exploratory Data Analysis*. Pearson, 1977. [p511]

X. Wang, K. A. Smith, and R. J. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006. [p514]

J. Whittaker, C. Whitehead, and M. Somers. The neglog transformation and quantile regression for the analysis of a large credit scoring database. *Journal of the Royal Statistical Society Series C*, 54(4): 863–878, 2005. [p511]

D. Wuertz, T. Setz, and Y. Chalabi. *timeSeries: Rmetrics - Financial Time Series Objects*, 2020. URL `https://cran.r-project.org/package=timeSeries`. R package version 3062.100. [p523]

Z. Yang. A modified family of power transformations. *Economics Letters*, 92(1):14–19, 2005. [p511]

G. Yapar. Modified simple exponential smoothing. *Hacettepe University Journal of Mathematics and Statistics*, 47(3):741–754, 2018. [p507, 522, 531]

G. Yapar, I. Yavuz, and H. T. Selamlar. Why and how does exponential smoothing fail? an in depth comparison of Ata-simple and simple exponential smoothing. *Turkish Journal of Forecasting*, 1(1): 30–39, 2017. [p507, 531]

G. Yapar, S. Capar, H. T. Selamlar, and I. Yavuz. Modified holt's linear trend method. *Hacettepe University Journal of Mathematics and Statistics*, 47(5):1394–1403, 2018. [p507, 531]

G. Yapar, H. T. Selamlar, S. Capar, and I. Yavuz. ATA method. *Hacettepe Journal of Mathematics and Statistics*, 48(6):1838–1844, 2019. [p507, 531]

I.-K. Yeo and R. A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000. [p510, 511]

T. E. Yilmaz, G. Yapar, and I. Yavuz. Comparison of Ata method and croston based methods on forecasting of intermittent demand. *Mugla Journal of Science and Technology*, 5(2):49–55, 2019. [p507]

Ali Sabri Taylan
Dokuz Eylul University
The Graduate School of Natural and Applied Sciences
Tinaztepe Campus Buca, Izmir, TURKEY
ORCiD 0000-0001-9514-934X alisabritaylan@gmail.com

Guckan Yapar
Dokuz Eylul University
The Graduate School of Natural and Applied Sciences
Tinaztepe Campus Buca, Izmir, TURKEY
ORCiD 0000-0002-0971-6676
guckan.yapar@deu.edu.tr

Hanife Taylan Selamlar
Dokuz Eylul University
The Graduate School of Natural and Applied Sciences
Tinaztepe Campus Buca, Izmir, TURKEY
ORCiD 0000-0002-4091-884X
hanife.taylan@deu.edu.tr

PASSED: Calculate Power and Sample Size for Two Sample Tests

by Jinpu Li, Ryan P. Knigge, Kaiyi Chen, Emily V. Leary

Abstract Power and sample size estimation are critical aspects of study design to demonstrate minimized risk for subjects and justify the allocation of time, money, and other resources. Researchers often work with response variables that take the form of various distributions. Here, we present an R package, **PASSED**, that allows flexibility with seven common distributions and multiple options to accommodate sample size or power analysis. The relevant statistical theory, calculations, and examples for each distribution using **PASSED** are discussed in this paper.

Introduction

Power and sample size estimation are critical aspects of study design to demonstrate minimized risk for subjects and justify the allocation of time, money, and other resources (Jones et al., 2003). A number of R packages for power analysis have been developed over the years. The **samplesize** (Scherer, 2016) package provides the calculation of sample size for the Student's t-test and the Wilcoxon-Mann Whitney test for categorical data. The **TrialSize** (Zhang et al., 2013) package implements the power analysis described in Chow et al. (2007), including power and sample size calculations for different study designs. Most recently, the **simglm** (LeBeau, 2019) package presents a simulation approach for power analysis that allows for the specification of missing data, unbalanced designs, and different random error distributions of generalized linear models.

Moreover, researchers often work with response variables that can take the form of a variety of distributions. For example, the proportion of thromboembolism after surgery in different treatment groups can be modeled using the binomial distribution or length of inpatient stay after an orthopedic procedure can be modeled using the Poisson distribution (Plessl et al., 2020). Some of the R packages or functions are designed to calculate the power and sample size for the variables following a certain distribution. The base package **stats** (R Core Team, 2016) provides such functions for normal (Gaussian) and binomially distributed variables, and the situations of unequal sample sizes are extended by packages **pwr** (Champely et al., 2017), **MESS**(Ekstrøm, 2012), **pwr2ppl**(Aberson, 2019), and **WebPower** (Zhang and Mai, 2018). The package **MKmisc** (Kohl, 2021) further adds a function for the comparison of negative binomial distributions. However, none of these packages provide a comprehensive power analysis toolkit capable of calculating power or sample sizes for the test of two-sample means or ratios when the responses have other common distributions (Table 1).

Package	Binomial	Normal	Negative Binomial	Geometric	Poisson	Beta	Gamma
PASSED	x	x	x	x	x	x	x
stats	x*	x*					
pwr	x	x**					
WebPower	x	x**					
MESS	x	x					
pwr2ppl	x	x					
MKmisc			x	x			

*: equal sample only; **: equal variance only.

Table 1: The comparison among **PASSED** and other available packages.

Here, we present an R package, **PASSED**, that performs power and sample size analyses for the following distributions: binomial, negative binomial, geometric, Poisson, normal (Gaussian), beta, and gamma distributions. Distributions, which had existing functions or R infrastructure for sample size and power calculations were included to streamline these calculations. However, calculations for the beta, Poisson, and gamma distributions were developed specifically for inclusion in **PASSED**. In the following sections, we will discuss the motivating examples, relevant statistical theory, and calculations for each distribution using **PASSED**.

PASSED: R Package Description

All functions in this package can be used to compute the power for a specific study design (e.g., given sample sizes) or to estimate specific parameter values (e.g., sample sizes) necessary to obtain a target power. The specific function of interest will depend on the type of outcome variable and the data distribution. All functions output an object of class `power.htest` that details the specified parameters of the test and the estimated parameter set as `NULL`.

Binomial

The binomial distribution is useful when modeling the number of successes in a sequence of independent and identically distributed Bernoulli trials. One example which uses data modeled using a binomial distribution is the proportion of blood transfusion that has occurred during surgery. The need for blood transfusion during surgery is an important consideration during surgical planning and particularly for surgical trials. LEITE et al. (2020) applied a logistic regression with binomial outcomes to model the rate of blood transfusions after the introduction of Tranexamic acid in knee arthroplasty.

Hypothesis Testing two-sample proportions is commonly considered in research designs when the outcome follows a binomial distribution. Let x_{ij} be a binary response from the jth subject in the ith group, $j = 1, ..., n_i$, $i = 1, 2$. It is assumed that x_{ij} are independent Bernoulli random variables with proportion p_i,

$$x_{ij} \sim Bernoulli(p_i)$$

Two hypothesis frameworks are considered for power and sample size calculations, which correspond to either a one-sided or two-sided test:

$$H_0 : p_1 = p_2 \ vs. \ H_a : p_1 \neq p_2 \ (two - sided)$$

or

$$H_0 : p_1 = p_2 \ vs. \ H_a : p_1 > (<) p_2 \ (one - sided)$$

Algorithm A binomial asymptotic test statistic was first proposed by Pearson (1900). Fleiss et al. (1980) provided an explicit formula to calculate the corresponding sample sizes for the test:

$$n_1 = \frac{[z_{\frac{\alpha}{2}} \sqrt{(r+1)\bar{p}\bar{q}} + z_\beta \sqrt{rp_1q_1 + p_2q_2}]^2}{rd^2} (two - sided)$$

or

$$n_1 = \frac{[z_\alpha \sqrt{(r+1)\bar{p}\bar{q}} + z_\beta \sqrt{rp_1q_1 + p_2q_2}]^2}{rd^2} (one - sided),$$

where $r = n_2/n_1$, $d = p_2 - p_1$, $q_1 = 1 - p_1$, $q_2 = 1 - p_2$, $\bar{p} = \frac{n_1p_1+n_2p_2}{n_1+n_2}$, $\bar{q} = 1 - \bar{p}$, and z_x denotes the probability that a standard normal deviate is greater than x. To obtain the power, this equation can be re-written as:

$$z_\beta = \frac{\sqrt{rn_1}|d| - z_{\frac{\alpha}{2}} \sqrt{(r+1)\bar{p}\bar{q}}}{\sqrt{rp_1q_1 + p_2q_2}} (two - sided)$$

$$z_\beta = \frac{\sqrt{rn_1}|d| - z_\alpha \sqrt{(r+1)\bar{p}\bar{q}}}{\sqrt{rp_1q_1 + p_2q_2}} (one - sided)$$

And, thus, the power can be derived as:

$$\text{Power} = \Pr\left(Z < \frac{z_{\frac{\alpha}{2}} \sqrt{(r+1)\bar{p}\bar{q}} - \sqrt{rn_1}|d|}{\sqrt{rp_1q_1 + p_2q_2}}\right) (two - sided)$$

$$\text{Power} = \Pr\left(Z < \frac{z_\alpha \sqrt{(r+1)\bar{p}\bar{q}} - \sqrt{rn_1}|d|}{\sqrt{rp_1q_1 + p_2q_2}}\right) (one - sided)$$

As a result, the target power, required sample sizes (n_1 and n_2), significance level (α), or the proportions (p_1 and p_2) can be obtained once all other remaining parameters are known (Fleiss et al., 1980). To optimize the sample size allocation, please refer to the discussion in Brittain and Schlesselman (1982).

Function The `power_Binomial()` function is useful when testing for differences among two sample proportions when the data follow a binomial distribution. This function uses the algorithm described above. The arguments for `power_Binomial()` are as follows:

```
power_Binomial(n1 = NULL, n2 = NULL, p1 = 0.5, p2 = 0.5,
               sig.level = 0.05, power = NULL, equal.sample = TRUE,
               alternative = c("two-sided", "one-sided"))
```

Sample sizes for each group are designated as `n1` and `n2`. If sample sizes for both groups are equal, the argument `equal.sample` should be set to `TRUE`, and only a value for `n1` is needed. If sample sizes are unequal, `equal.sample` should be set to `FALSE`, and values for both `n1` and `n2` must be specified. When estimating other parameters, the target power must be set with `power`. The significance level is set with `sig.level` and has a default value of 0.05. The probability of success for each group is indicated as `p1` and `p2`, respectively, with 0.5 as the default value for both. Only one of the parameters of `n1`, `n2`, `p1`, `p2`, `power`, or `sig.level` can be set as `NULL`. The parameter set as `NULL` will be estimated based on the other parameter values. The argument `alternative` specifies the alternative hypothesis as either `"two.sided"` (default) or `"one.sided"`.

The `power_Binomial()` function returns the same results as `stats::power.prop.test()` in the equal sample scenario. It also allows power calculations with unequal sample sizes, and the results are identical to `MESS::power_prop_test()`.

Negative Binomial

The negative binomial distribution can be used to model the number of successes in a sequence of independent and identically distributed Bernoulli trials before a specified number of failures occurs. Gates et al. (2020) analyzed the probability of positive intraoperative cultures in a population of patients with a history of prior ipsilateral shoulder surgery. The probability of the total number of positive tissue cultures was modeled using a generalized negative binomial mixed model with maximum likelihood estimation and robust standard errors. Using this negative binomial framework, the appropriate sample size and power for such a study can be obtained using the method outlined below.

Hypothesis Consider a sequence of adverse events. Let x_{ij} be the number of events during time t_i from the jth subject in the ith group, $j = 1, ..., n_i$, $i = 1, 2$. Assuming that x_{ij} are negative binomial random variables with a mean μ_{ij} and parameter θ ($\theta > 0$), the probability function of x_{ij} is

$$P\left(x_{ij}\right) = \frac{\Gamma\left(\theta + x_{ij}\right)}{\Gamma\left(\theta\right) x_{ij}!} \left(\frac{\mu_{ij}}{\theta + \mu_{ij}}\right)^{x_{ij}} \left(\frac{\theta}{\theta + \mu_{ij}}\right)^{\theta}, \tag{1}$$

where $n!$ denotes the product of the integers from 1 to n and $\Gamma(\cdot)$ is the gamma function (Zhu and Lakkis, 2014).

To model the negative binomial outcomes, Hilbe (2011) introduced the negative binomial regression. Zhu and Lakkis (2014) then presented a hypothesis test comparing two negative binomial distributed samples using negative binomial regression, and this is the method used here. In negative binomial regression, μ_{ij} can be modeled as

$$log(\mu_{ij}) = log(t_i) + \beta_0 + \beta_1 G_{ij},$$

where G_{ij}, the group indicator for subject j, is equal to 0 if $i = 1$ for group 1 and is equal to 1 if $i = 2$ for group 2. Let r_1 and r_2 be the mean rates of events per time unit for groups 1 and 2, which can be expressed as $r_1 = e^{\beta_0}$ and $r_2 = e^{\beta_0+\beta_1}$. Then $r_2/r_1 = e^{\beta_1}$ can be easily obtained (Zhu and Lakkis, 2014).

To compute the power of the test or determine parameters to obtain target power, two hypothesis frameworks are considered which correspond to either a one-sided or two-sided test:

$$H_0 : \frac{r_2}{r_1} = 1 \ vs. \ H_a : \frac{r_2}{r_1} \neq 1 \ (two - sided)$$

or

$$H_0 : \frac{r_2}{r_1} = 1 \ vs. \ H_a : \frac{r_2}{r_1} > (<) 1 \ (one - sided)$$

Algorithm The power and sample size calculation algorithms were developed by Zhu and Lakkis (2014) based on the asymptotic normality of the maximum likelihood estimation of β_1. The power can be calculated as:

$$power = \Phi\left(\frac{\sqrt{n_1}\left|\log\left(\frac{r_2}{r_1}\right)\right| - z_{\frac{\alpha}{2}}\sqrt{V_0}}{\sqrt{V_1}}\right) \quad (two - sided)$$

or

$$power = \Phi\left(\frac{\sqrt{n_1}\left|\log\left(\frac{r_2}{r_1}\right)\right| - z_{\alpha}\sqrt{V_0}}{\sqrt{V_1}}\right) \quad (one - sided),$$

where V_0 and V_1 are the estimates of variance for $\hat{\beta}_1$ by n_1 under H_0 and H_a,

$$V_0 = \frac{1}{t_i}\left(\frac{1}{\tilde{r}_1} + \frac{n_1}{n_2\tilde{r}_2}\right) + \frac{(n_1+n_2)}{\theta n_2}$$

$$V_1 = \frac{1}{t_i}\left(\frac{1}{r_1} + \frac{n_1}{n_2 r_2}\right) + \frac{(n_1+n_2)}{\theta n_2},$$

and $\tilde{r}_i$, $i = 1, 2$ denotes the estimation of the event rate under H_0 in each group. Zhu and Lakkis (2014) provided three approaches to estimating $\tilde{r}_i$ under H_0:

Approach 1: using event rate of group 2 (reference group rate)

$$V_0 = \frac{1}{t_i}\left(\frac{1}{r_2} + \frac{n_1}{n_2 r_2}\right) + \frac{(n_1+n_2)}{\theta n_2};$$

Approach 2: using true rates

$$V_0 = \frac{1}{t_i}\left(\frac{1}{r_1} + \frac{n_1}{n_2 r_2}\right) + \frac{(n_1+n_2)}{\theta n_2};$$

Approach 3: using maximum likelihood estimation

$$V_0 = \frac{1}{t_i}\left(\frac{1}{\frac{n_1 r_1 + n_2 r_2}{n_1+n_2}} + \frac{n_1}{n_2\frac{n_1 r_1 + n_2 r_2}{n_1+n_2}}\right) + \frac{(n_1+n_2)}{\theta n_2}.$$

Function The function `power_NegativeBinomial()` is useful when developing a study design to compare differences in rates when the data follow a negative binomial distribution. Calculations for this function are based on Zhu and Lakkis (2014). The following arguments are used:

```
power_NegativeBinomial(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
                       mu1 = NULL, mu2 = NULL, duration = 1, theta = NULL,
                       equal.sample = TRUE,
                       alternative = c("two-sided", "one-sided"), approach = 3)
```

The sample size for each group is specified as `n1` and `n2`, both with default values of `NULL`. When sample sizes are equal, `equal.sample` can be set to `TRUE`, and only `n1` must be specified. Otherwise `equal.sample` is set to `FALSE` and values must be input for both `n1` and `n2`. The `power` argument is set to `NULL` unless the target power is specified here and another parameter is set as `NULL` to be estimated. The significance level for the test is set by `sig.level` with a default value of 0.05. The expected rates of events per unit time for each group are denoted as `mu1` and `mu2`, respectively, with the average treatment duration set by `duration` (default value of 1). `Theta` indicates the θ parameter of the negative binomial distribution, as noted above. The argument `alternative` specifies the alternative hypothesis as either `"two.sided"` (default) or `"one.sided"`. Lastly, the argument `approach` can be set as either "1", "2", or "3" (default). These values indicate the selection of one of three procedures for estimating the variance under the null hypothesis for the sample size formula and correspond with Approach 1 (reference group rate), Approach 2 (true rates), and Approach 3 (maximum likelihood estimation) described above. The obtained results match other functions in R such as `MKmisc::power.nb.test()`.

Geometric

The geometric distribution can be used to examine the probability of success given a limited number of trials and is considered a special case of the negative binomial distribution. For example, in baseball, the probability of a batter earning a hit before striking out can be compared to that of another batter, using a geometric distribution.

Hypothesis Let x_{ij} be the number of events during time t_i from the jth subject in the ith group. Assuming that x_{ij} are geometric random variables with a mean μ_{ij}, the probability function of x_{ij} is

$$P\left(x_{ij}\right) = \left(\frac{\mu_{ij}}{1+\mu_{ij}}\right)^{x_{ij}} \left(\frac{1}{1+\mu_{ij}}\right).$$

Referring to Equation 1, this is a special case of the negative binomial where $\theta = 1$. Similarly, μ_{ij} can be modeled as shown in the Section Negative Binomial,

$$log(\mu_{ij}) = log(t_i) + \beta_0 + \beta_1 G_{ij}.$$

The hypotheses and calculations follow as previously shown in the Section Negative Binomial.

Algorithm The power and sample size calculation formula are the same as the Section Negative Binomial, with $\theta = 1$.

Function The function `power_Geometric()` applies the same algorithm as the function `power_NegativeBinomial()`, with the same arguments, where the parameter `theta` is set as 1. See `power_NegativeBinomial()` for more details.

```
power_Geometric(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05, mu1 = NULL,
                mu2 = NULL, duration = 1, equal.sample = TRUE,
                alternative = c("two-sided", "one-sided"), approach = 3)
```

Poisson

The Poisson distribution can be used to model the number of events occurring in a fixed interval of time or space. In healthcare, length of stay (LOS) is one of many important considerations for interventions, particularly when inpatient hospital stay may vary among treatments. LOS, or other count measurements important to the research study, can be modeled using a Poisson distribution. Plessl et al. (2020) used a Poisson framework to compare LOS for those who were treated with rapid recovery protocols versus standard recovery protocols after total knee arthroplasty. This example can be expanded to the general case as follows.

Hypothesis Let x_{ij} be the number of events during the necessary study time t_i from the jth subject in the ith treatment group, $j = 1, ..., n_i$, $i = 1, 2$. This situation is commonly referred to as the equal sampling frame approach (Hutchinson and Holtman, 2005). It is assumed that x_{ij} are Poisson random variables with rate λ_i such that the probability function of x_{ij} is

$$P\left(x_{ij}\right) = \frac{t_i \lambda_i e^{t_i \lambda_i}}{x_{ij}!},$$

where $i = 1, 2$. Then, the total number of events in each group, denoted as X_1 and X_2, also follow a Poisson distribution:

$$X_i \sim Poisson(\lambda_i t_i n_i)$$

Four methods have previously been proposed to test the equality of two Poisson rates (Shiue and Bain, 1982; D. Huffman, 1984; Thode, 1997; Gu et al., 2008). The method utilized in the **PASSED** package was proposed by Gu et al. (2008), which considers the ratio of two Poisson rates, R, a pre-specified positive number. The asymptotic test is as follows:

$$H_0 : \lambda_2/\lambda_1 = R \; vs. \; H_a : \lambda_2/\lambda_1 = R' \neq R(two - sided)$$

or

$$H_0 : \lambda_2/\lambda_1 = R \; vs. \; H_a : \lambda_2/\lambda_1 = R' > R(one - sided)$$

Algorithm The following formula is used in the **PASSED** package and the details of the derivation are provided in the Appendix.

$$n_1 = \frac{\left(\frac{z_{1-\frac{\alpha}{2}} C + z_{power} D}{A}\right)^2 - \frac{3}{8}}{\lambda_1 t_1} (two - sided)$$

or

$$n_1 = \frac{(\frac{z_{1-\alpha}C + z_{power}D}{A})^2 - \frac{3}{8}}{\lambda_1 t_1} (one - sided)$$

where $A = 2(1 - \sqrt{\frac{R}{R'}})$,
$B = \lambda_1 t_1 n_1 + 3/8$,
$C = \sqrt{\frac{R+d}{R'}}$,
$D = \sqrt{\frac{R'+d}{R'}}$
$d = t_1/t_2$.

Function The `power_Poisson()` function is designed to compute the power or estimate parameters to obtain a target power when testing for a ratio of two Poisson rates. This function applies the asymptotic tests based on normal approximations developed by Gu et al. (2008). The arguments for `power_Poisson()` are as follows:

```
power_Poisson(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
              lambda1 = NULL, lambda2 = NULL, t1 = 1, t2 = 1, RR0 = 1,
              equal.sample = TRUE, alternative = c("two.sided", "one.sided"))
```

Sample sizes for each group are set with `n1` and `n2`. If sample sizes for both groups are equal, the argument `equal.sample` should be set to `TRUE`, and only a value for `n1` needs to be specified. If sample sizes are unequal, `equal.sample` should be set to `FALSE`, and values for both `n1` and `n2` must be specified. The target power of the test is set with `power`, and the significance level is set with `sig.level` (default value of 0.05). The expected rates of events per unit time for each group are denoted as `lambda1` and `lambda2`, respectively, with the average treatment duration set by `t1` and `t2` (default value of 1). Only one of the parameters of `n1`, `n2`, `lambda1`, `lambda2`, or `power` can be set as `NULL` for the function to run. The parameter set as `NULL` will be estimated based on the other parameter values. `t1` and `t2` refer to the specified interval of time (or space) where the events occur. The rate ratio from the null hypothesis is specified as `RR0`. It should be set to 1 when testing for equal Poisson rates. The argument `alternative` specifies the alternative hypothesis as either `"two.sided"` (default) or `"one.sided"`.

For the example in Gu et al. (2008), which aims at testing if the risk of coronary heart disease is greater for those with postmenopausal hormone use (RR0 = 1), the event rates for those with and without hormone use are assumed to be 0.2000 and 0.0005 (lambda2 = 0.0020, lambda1 = 0.0005), respectively, during a 2-year time period (t1 = t2 = 2). Given the sample size for each group as 4295 and 8590 (n2 = 4295, n1 = 8590) [1], the power under a significance level of 0.05 can be calculated as follows:

```
power_Poisson(n1 = 8590, n2 = 4295, power = NULL, sig.level = 0.05,
                      lambda1 = 0.0005, lambda2 = 0.0020, t1 = 2, t2 = 2, RR0 = 1,
                      equal.sample = FALSE, alternative = "one.sided")
```

The estimated power is 0.9000147, which matches the results in Gu et al. (2008).

Normal

The normal distribution is widely used in the natural and social sciences. Age is a common demographic variable recorded during patient care and typically follows a normal distribution. Many surgeons consider demographic variables to evaluate the possible risks of a surgical procedure and assess optimal treatment options for patients. Luan et al. (2020) aimed to identify patients who were suitable for kinematic or mechanical alignment of the knee. To compare these groups, Luan et al. (2020) used the student t-test to compare normally distributed age.

Hypothesis T-tests are widely used to compare two sample means when the data has a normal distribution (Cressie and Whitford, 1986). Let x_{ij} be a continuous response from the jth subject in the ith group, $j = 1, ..., n_i$, $i = 1, 2$. It is assumed that x_{ij} are independent, normal random variables with mean μ_i and variance σ_i^2:

$$x_{ij} \sim Normal(\mu_i, \sigma_i^2),$$

[1]the sample sizes are corrected in NCSS Software Manuals 2020 Page 437-14, "Tests for the Ratio of Two Poisson Rates"

then the probability density function of x_{ij} is:

$$f\left(x_{ij}\right) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\left(\frac{x_{ij}-\mu_i}{\sigma_i}\right)^2},$$

where $i = 1, 2$. It can be shown that the mean of each group, denoted as $\bar{x}_1$ and $\bar{x}_2$, also follows a Normal distribution:

$$\bar{x}_i \sim Normal(\mu_i, \frac{\sigma_i^2}{n_i})$$

To compute the power for a hypothesis test or determine parameters to obtain a target power for hypothesis, the following two scenarios are considered:

$$H_0 : \mu_1 = \mu_2 \; vs. \; H_a : \mu_1 \neq \mu_2 \; (two - sided)$$

or

$$H_0 : \mu_1 = \mu_2 \; vs. \; H_a : \mu_1 > (<) \mu_2 \; (one - sided)$$

Algorithm Based on the work of Ekstrøm (2012), in the **PASSED** package, the user can define the sample sizes (n_1 and n_2) and standard deviations (σ_1 and σ_2) of each group directly, rather than set the size ratio (n_2/n_1) and standard deviation ratio (σ_2/σ_1). To optimize sample size allocation, please refer to the discussion in Jan and Shieh (2011).

Function The `power_Normal()` function is useful for developing a study design to test for differences between mean values of two groups when the data follow a normal distribution. This function performs the same operations as `pwr.t.test` in the **pwr** package (Champely et al., 2017) but allows for additional parameter modifications. In particular, this function allows for specifying unequal sample sizes and standard deviations across groups. The arguments for `power_Normal()` are as follows:

```
power_Normal(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
             delta = NULL, sd1 = 1, sd2 = 1, equal.sample = TRUE,
             alternative = c("two-sided", "one-sided"),
             type = c("two-sample", "one-sample", "paired"),
             df.method = c("welch", "classical"), strict = FALSE)
```

Sample sizes for each group are set with `n1` and `n2`. If sample sizes for both groups are equal, the argument `equal.sample` should be set to `TRUE`, and only a value for `n1` needs to be specified. If sample sizes are unequal, `equal.sample` must be set to `FALSE`, and values for both `n1` and `n2` must be specified. The target power of the test is set with `power`, and the significance level is set with `sig.level` (default value of 0.05). `delta` indicates the difference in means between the two groups, and `sd1` and `sd2` denote the standard deviations for each group. A default value of 1 is indicated for both `sd1` and `sd2`. The default values for `n1`, `n2`, `power`, and `delta` are `NULL`, whereas `sd1`, `sd2`, and `sig.level` have non-NULL default values. Only one of the parameters can be set as `NULL`. The parameter set as `NULL` will be estimated based on the other parameter values. The type of t-test is indicated by `type` and set as `"two.sample"` (default), `"one.sample"`, or `"paired"`. `alternative` specifies the alternative hypothesis as either `"two.sided"` (default) or `"one.sided"`. Lastly, `df.method` indicates the method for calculating the degrees of freedom as either `"welch"` (default) or `"classical"`. Note that setting `strict` as `TRUE` would be applied only in the two-sided case, when the probability of rejection in the opposite direction of the true effect is included, i.e., the alternative hypothesis of the two-sided t-test is $\mu_1 \neq \mu_2$ rather than $\mu_1 > (<) \mu_2$.

The `power_Normal()` function produces the same results as `stats::power.t.test()` for the equal sample size scenario. It also allows power calculations with unequal sample sizes and unequal variances. The results match other functions in R such as `MESS::power_prop_test()` and `pwr::pwr.t2n.test()`.

Beta

The beta family of continuous probability distributions is ideal for modeling data with right or left skewness and allows the probability density to assume a variety of shapes through two shape parameters (Gupta and Nadarajah, 2004). Disease status is often measured with bounded outcome scores, which take values on a finite range. The distribution of such data is often skewed, rendering the standard analysis methods assuming a normal distribution inappropriate (Hu et al., 2020), and thus, a beta distribution can be utilized. This scenario can be generalized as follows.

Hypothesis Suppose a sequence of random responses, x_{ij} from the jth subject in the ith group, takes the form of a continuous proportion that follows a beta distribution, $x_{ij} \sim Beta(a_i, b_i)$, where $j = 1, ..., n_i$, $i = 1, 2$. The probability density function of x_{ij} is:

$$f(x_{ij}) = \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} x_{ij}^{a_i - 1}(1 - x_{ij})^{b_i - 1},$$

where $0 \leq x_{ij} \leq 1$, $a_i > 0$, $b_i > 0$, and $i = 1, 2$. When analyzing continuous proportions as a response variable, the standard shape parameters of a beta density, a_i and b_i, are often not directly observable. Ferrari and Cribari-Neto (2004) developed a class of beta regression models which utilize an alternative parameterization of the beta density function based on the mean, μ_i, and an unknown precision parameter, ϕ_i. Suppose $\mu_i = a_i/(a_i + b_i)$ and $\phi_i = a_i + b_i$, then the beta density function can be expressed in terms of μ_i and ϕ_i as below:

$$f(x_{ij}) = \frac{\Gamma(\phi_i)}{\Gamma(\mu_i\phi_i)\Gamma((1 - \mu_i)\phi_i)} x_{ij}^{\mu_i\phi_i - 1}(1 - x_{ij})^{(1-\mu_i)\phi_i - 1};$$

For beta regression, μ_i can be modeled as

$$g(\mu_i) = \beta_0 + \beta_1 G_{ij},$$

where G_{ij}, the group indicator for subject j, is equal to 0 if $i = 1$ for group 1 and is equal to 1 if $i = 2$ for group 2, and $g(\cdot)$ denotes the link function. The **PASSED** package includes the capability for the following link functions and their respective forms:

- Logit: $g(\mu) = log\left[\frac{\mu}{(1-\mu)}\right]$
- Probit: $g(\mu) = \Phi^{-1}(\mu)$
- Complementary log-log: $g(\mu) = log[-log(1 - \mu)]$
- Log: $g(\mu) = log(\mu)$
- Log-log: $g(\mu) = -log[-log(\mu)]$

The equality of means μ_i is equivalent to $\beta_1 = 0$. The objective is to compute the power of the test or determine minimum sample sizes to obtain a target power for the needed hypothesis. A two-sided hypothesis framework is considered for power and sample size calculations:

$$H_0 : \mu_1 - \mu_2 = 0 \; vs. \; H_a : \mu_1 - \mu_2 \neq 0$$

Algorithm The mean and variance of x_{ij}, denoted as μ_i and σ_i^2, can be obtained using:

$$\mu_i = \frac{a_i}{a_i + b_i}$$

and

$$\sigma_i^2 = \frac{a_i b_i}{(a_i + b_i)^2(a_i + b_i + 1)}.$$

Incorporating the definition of the precision parameter ϕ_i, the following equations can be derived:

$$a_i = \mu_i\phi_i = \mu_i\left(\frac{\mu_i(1 - \mu_i)}{\sigma_i^2} - 1\right); \tag{2}$$

$$b_i = (1 - \mu_i)\phi_i = (1 - \mu_i)\left(\frac{\mu_i(1 - \mu_i)}{\sigma_i^2} - 1\right). \tag{3}$$

To calculate power, a simulation approach is used. Parameters μ_i and ϕ_i are first estimated using the given mean and variance, then they are used to obtain the original beta parameters, a_i and b_i, following Equations 2 and 3. The response variable is simulated for each distribution, $Beta(a_1, b_1)$ and $Beta(a_2, b_2)$, with the given sample size. If any simulated response is equal to zero or one, the following transformation is applied to each response value from both distributions: $(x(n - 1) + 0.5)/n$, where x is the response value and n is the sample size (Smithson and Verkuilen, 2006).

Values for the simulated response from both distributions are merged together, along with the group indicator (0 for group 1 and 1 for group 2). Subsequently, a beta regression model is built using the specified link type (Cribari-Neto and Zeileis, 2010). A Wald test is performed on the simulated model, testing the null hypothesis that β_1 is equal to 0. The p-values are recorded for each test and the simulation is repeated M times. The power is calculated as:

$$\text{power} = \frac{\text{Number of p-values less than 0.05}}{M}.$$

Let ss denote sample size, then the generic power/sample size relationship can be formally expressed as:

$$\text{power} = \text{f(ss)}$$

Assuming the response variable follows a beta distribution, $f(\cdot)$ is continuous on the interval (0, 1) and increases monotonically. Consequently, the `power_Beta` function uses the bisection method to obtain the minimum sample size, ss_0, through a sequence of steps for each iteration (Chernick and Liu, 2002). For each target power, $power_0$, upper and lower sample size bounds, ss_u and ss_l, which satisfy $f(ss_l) < power_0 < f(ss_u)$ are established using a two-sample t-test performed with the base function `power.t.test` (R Core Team, 2016). Although `power.t.test` assumes normality, it is useful to generate starting values for ss_u and ss_l.

The sequence of steps for each iteration is as follows:

1. Compute the midpoint $ss_{mid} = floor(\frac{ss_l+ss_u}{2})$ of interval $[ss_l, ss_u]$. $floor(\cdot)$ denotes retaining the integer part of a number.
2. Calculate power at the midpoint, ss_{mid}, using the simulation described for the power calculations above.
3. If $f(ss_{mid}) \geq power_0$ and $ss_{mid} - ss_l \leq 1$, then return ss_{mid} and stop iterating.
4. Examine the sign of $f(ss_{mid}) - power_0$. If negative, then replace ss_l with ss_{mid}, otherwise replace ss_u with ss_{mid} so that $f(ss_l) < power_0 \leq f(ss_u)$.

Repeat the process until iteration stops. The output minimum sample size, ss_0, is the minimum integer such that $f(ss_0) \geq power_0$.

Function The `power_Beta()` function is framed to test differences between mean values for two groups, assuming the response variable follows a beta distribution in each group. It can be used to compute the power or to estimate the required sample sizes to obtain a target power. In particular, this function allows for specifying unequal sample sizes and standard deviations across groups. The arguments for `power_Beta()` are as follows:

```
power_Beta(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
           mu1 = NULL, sd1 = NULL, mu2 = NULL, equal.sample = TRUE,
           trials = 100, equal.precision = TRUE, sd2 = NULL,
           link.type = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"))
```

Sample sizes for each group are set with `n1` and `n2`. If sample sizes for both groups are equal, the argument `equal.sample` should be set to `TRUE`, and only a value for `n1` or `power` needs to be specified. If sample sizes are unequal, `equal.sample` should be set to `FALSE`, and values for both `n1` and `n2` must be specified. The target power of the test is set with `power`, and the significance level is set with `sig.level` (default value of 0.05). Only one of the parameters of `n1`, `n2`, or `power` can be `NULL`. The mean and standard deviation for the null distribution are denoted by `mu1` and `sd1`. Analogously, the mean and standard deviation for the alternative distribution can be specified by `mu2` and `sd2`. Note that `equal.precision=FALSE` should be used to set the standard deviation for the alternative distribution, meaning the precision parameters are assumed to be unequal. Otherwise, option `sd2` would be ignored. The option `trials` indicates the number of trials in the simulation. A default number of trials (i.e., 100) is recommended to get a rough estimate of other parameters (e.g., `sd2`), since the computational time is dependent upon the number of trials in the simulation. Once an appropriate range of other values is determined, the number of trials should be increased (e.g., trials=1000) to calculate precise power and sample size estimates. The default link function is the logit link but can be changed using `link.type` with the following options: `"logit"`, `"probit"`, `"cloglog"`, `"log"`, `"loglog"`, to denote the logit, probit, complementary log-log, log, and log-log link functions, respectively.

Gamma

The gamma distribution is widely used to fit lifetime data because its flexibility in shape can vary from extremely positively skewed to almost symmetric (Casella and Berger, 2002). Hong et al. (2020) provide an example of modeling data using the gamma distribution to test the association of patient-provider cost discussion with out-of-pocket spending among cancer survivors. The data (i.e., out-of-pocket spending in cancer care) have an obvious skewness which is not normally distributed; and therefore, the two-sample t-test is not suitable for this purpose. Alternatively, gamma models can be used to test the difference of average total out-of-pocket spending between the patients with and without a patient-provider cost discussion.

Hypothesis Currently, there is no explicit formula to calculate the power comparing two gamma random variables. Let x_{ij} be a continuous response from the *jth* subject in the *ith* group, $j = 1, ..., n_i$, $i = 1, 2$. It is assumed that x_{ij} are gamma random variables with scale λ_i and shape δ_i so that the probability density function can be written as

$$f\left(x_{ij} = x\right) = \left(\frac{\lambda_i}{\Gamma\left(\delta_{\mathrm{i}}\right)}\right) x^{\delta_i - 1} e^{-\lambda_i x}.$$

The mean of $Gamma(\lambda_i, \delta_i)$ can be obtained using $\mu_i = \delta_i / \lambda_i$. Shiue and Bain (1983) developed a test of two equal gamma means with unknown common shape parameter, such that

$$H_0 : \mu_1 = \mu_2 = \mu.$$

This can be re-written as $H_0 : \delta = \lambda_1 \mu = \lambda_2 \mu$, some $\delta > 0$. This can then be tested using an F distribution based on the ratio of the mean of a random sample from two gamma distributions. In 1988, Shiue et al. (1988) extended this to the unknown and unequal shape parameter scenarios. However, this extension can be slightly conservative and problematic for smallscale parameters. More recently, Chang et al. (2011) provided a computational approach using a variant of the parametric bootstrap method, used here, in which the shape parameters are completely unknown and unequal. In this characterization, the hypothesis is two-sided and is of the form $H_0 : \delta_i = \lambda_i \mu$, some $\mu > 0$ or equivalently, for two means,

$$H_0 : \frac{\delta_1}{\lambda_1} = \frac{\delta_2}{\lambda_2} \; vs. \; H_a : \frac{\delta_1}{\delta_1} \neq \frac{\delta_2}{\lambda_2}.$$

This can be expressed as a scalar value function, η, such that

$$H_0^* : \eta = \sum_{i=1}^{2} \left(\beta_i - \bar{\beta}\right)^2 = 0 \; vs. \; H_a^* : \eta > 0,$$

where $\beta_i = \ln\left(\mu_i\right)$ and $\bar{\beta} = \sum_{i=1}^{2} \frac{\beta_i}{2}$.

Algorithm The power and sample size calculation algorithm adapted for **PASSED** was developed by Chang et al. (2011). This computational approach performs best when the restricted maximum likelihood estimate of η behaves as approximately normal or as a sum of squared normals.

Function The `power_Gamma()` function is used to compute the power or estimate sample sizes to obtain a target power when testing for differences among two sample means when the data follow a gamma distribution. This function used a parametric bootstrap method addressed by Chang et al. (2011). The arguments for `power_Gamma()` are as follows:

```
power_Gamma(n1 = NULL, n2 = NULL, power = NULL, sig.level = 0.05,
            mu1 = NULL, mu2 = NULL, gmu1 = NULL, gmu2 = NULL,
            trials = 100, M = 10000, equal.sample = TRUE, equal.shape = NULL)
```

Sample sizes for each group are set with `n1` and `n2`. If sample sizes for both groups are equal, the argument `equal.sample` should be set to `TRUE`, and only a value for `n1` or `power` needs to be specified. If sample sizes are unequal, `equal.sample` should be set to `FALSE`, and values for both `n1` and `n2` must be specified. The target power of the test is set with `power`, and the significance level is set with `sig.level` (default value of 0.05). Only one of the parameters of `n1`, `n2`, or `power` can be set as `NULL`. The parameter set as `NULL` will be estimated based on the other parameter values. The arithmetic means for each group are indicated by `mu1` and `mu2`, while `gmu1` and `gmu2` denote the geometric mean for each group, respectively. Option `trials` specifies the number of trials in the simulation, and the number of generated samples in every single trial is identified by `M`. A small number of trials (e.g., using the default value 100) is recommended to get a rough estimate of power or sample size since the computational time is dependent upon the number of trials in the simulation. To obtain a reasonable result, a greater value (e.g., 10000) should be used for both `trials` and `M`. The assumption of equal shape parameters should be tested before the comparison of two sample means if `equal.shape` is set as `NULL` (default value is `NULL`). Otherwise, the test to determine equal shape is skipped (when `equal.shape` is set to be `TRUE` or `FALSE`).

For example, Schickedanz and Krause (1970) presented the weekly rainfall data for the seasons of fall and winter. The arithmetic/geometric means are 0.3684/0.2075 for winter (n = 57) and 0.7635/0.3630 for fall (n = 51). Using a significance level of 0.05, the power can be calculated as follows:

```
set.seed(1)
power_Gamma(n1 = 57, n2 = 51, power = NULL, sig.level = 0.05,
            mu1 = 0.3684, mu2 = 0.7635, gmu1 = 0.2075, gmu2 = 0.3630,
            trials = 100, M = 1000)
```

The estimated power is 1.00, which matches the result in Schickedanz and Krause (1970).

Application of PASSED

In this section, we provide an example power analysis and sample size calculation implemented with **PASSED**. We propose a hypothetical study to test an intervention protocol designed to reduce the percentage of residents at nursing facilities who develop new or worsening pressure ulcers, known as bedsores.

The Skilled Nursing Facility Quality Reporting Program (SNF-QRP) provider dataset contains information on pressure ulcer rates among nursing home facilities across the US. In this scenario, half of the participating nursing homes will implement the intervention protocol (treatment group), and the other half will constitute a control group, without a change in protocol, to determine if the new intervention reduces rates of pressure ulcers. We consider the following hypotheses for the study:

- H_0: There is no difference in pressure ulcer rates among nursing home facilities between control and treatment groups.
- H_a: There is a difference in pressure ulcer rates among nursing home facilities between control and treatment groups.

Sample Size Determination

In this example, we use the mean and standard deviation of the SNF-QRP variable, "percentage of SNF residents with pressure ulcers that are new or worsened" for the control group `mu1` and `sd1`, 0.0174 and 0.0211, respectively. A 25% decrease in the proportion of patients that develop new or worsening pressure ulcers is considered significant and results in the target alternative mean, `mu2`, equal to 0.0131. To determine the appropriate number of facilities necessary in the control and treatment groups, we first use `power_Beta` to estimate the minimum sample size with target power equal to 0.8. The `power_Beta` is chosen because this proportion is defined on the interval [0, 1] and right-skewed. The default value of `link.type` is used, `trials` is set at 1000, and equal precision in the control and treatment groups is assumed. This analysis can be fine-tuned through additional iterations of `power_Beta` by modifying the number of trials. The output is given below:

```
library(PASSED)
set.seed(1)
power_Beta(mu1 = 0.0174, sd1 = 0.0211, mu2 = 0.0131, power = 0.8,
           link.type = "logit", trials = 1000, equal.precision = TRUE)

     Two-sample Beta Means Tests (Equal Sizes) (logit link, equal precision)

              N = 151
            mu1 = 0.0174
            mu2 = 0.0131
            sd1 = 0.0211
      sig.level = 0.05
          power = 0.826

NOTE: N is number in *each* group
```

The obtained result indicates that 302 nursing home facilities (151 facilities for each group) are necessary to demonstrate the difference between pressure ulcer rates among the control and treatment groups, with a significance level of 0.05 and power of 0.80.

Comparison with T-Test

To further assess the appropriate number needed in the control and treatment groups, we then use 0.0120 to 0.0140 to evaluate a range of target means that encompass the target's alternative mean of 0.0131, with expected sample sizes of over 100 nursing homes per group. As a comparison, we also calculate the power using a two-sided t-test under the same scenario, using the function `power_Normal`. The true difference in means, `delta`, is set as the difference of `mu1` and `mu2`, and the alternative standard deviation is assumed to be equal to `sd1`. The output for this example is displayed below, assuming equal precision.

```
# Set seed for the simulation below
set.seed(1)
Ex1 <- mapply(
  function(mu2, sample_size){
    Betapower <- power_Beta(mu1 = 0.0174, sd1 = 0.0211,
                            mu2 = mu2, n1 = sample_size,
                            link.type = "logit", trials = 1000,
                            equal.precision = TRUE)
    Normalpower <- power_Normal(delta = (0.0174 - mu2), n1 = sample_size,
                               sd1 = 0.0211, sd2 = 0.0211)
    return(c(Betapower$power,
             round(Normalpower$power,3),
             sample_size,
             mu2,
             0.0174))
  },
  # Range of mu2 was set as [0.0120, 0.0140] by 0.0010
  rep(seq(0.0120, 0.0140, 0.0010), 5),
  # Range of sample size was set as [100, 200] by 25
  rep(seq(100, 200, 25), rep(3, 5))
)
# Reform the output
Ex1 <- as.data.frame(t(Ex1))
# Set column names
colnames(Ex1) <- c("Power (Beta)",
                   "Power (Normal)",
                   "Sample Size",
                   "mu2",
                   "mu1")
# Display the results
Ex1

   Power (Beta) Power (Normal) Sample Size   mu2    mu1
1         0.813          0.437         100 0.012 0.0174
2         0.623          0.311         100 0.013 0.0174
3         0.435          0.204         100 0.014 0.0174
4         0.891          0.522         125 0.012 0.0174
5         0.743          0.375         125 0.013 0.0174
6         0.488          0.245         125 0.014 0.0174
7         0.954          0.598         150 0.012 0.0174
8         0.821          0.436         150 0.013 0.0174
9         0.576          0.285         150 0.014 0.0174
10        0.979          0.665         175 0.012 0.0174
11        0.872          0.494         175 0.013 0.0174
12        0.609          0.324         175 0.014 0.0174
13        0.986          0.723         200 0.012 0.0174
14        0.914          0.548         200 0.013 0.0174
15        0.708          0.362         200 0.014 0.0174
```

When equal precision cannot be assumed, `equal.precision` is set to `FALSE`, and an input value for `sd2` is required. To demonstrate unequal precision, the previous example is rerun with `equal.precision=FALSE` and `sd2=0.03`. The output is provided below.

```
# Set seed for the simulation below
```

```
set.seed(1)
Ex2 <- mapply(
  function(mu2, sample_size){
    Betapower <- power_Beta(mu1 = 0.0174, sd1 = 0.0211, sd2 = 0.030,
                            mu2 = mu2, n1 = sample_size,
                            link.type = "logit", trials = 1000,
                            equal.precision = FALSE)
    Normalpower <- power_Normal(delta = (0.0174 - mu2), n1 = sample_size,
                            sd1 = 0.0211, sd2 = 0.030)
    return(c(Betapower$power,
             round(Normalpower$power,3),
             sample_size,
             mu2,
             0.0174))
  },
  # Range of mu2 was set as [0.0120, 0.0140] by 0.0010
  rep(seq(0.0120, 0.0140, 0.0010), 5),
  # Range of sample size was set as [100, 200] by 25
  rep(seq(100, 200, 25), rep(3, 5))
)
# Reform the output
Ex2 <- as.data.frame(t(Ex2))
# Set column names
colnames(Ex2) <- c("Power (Beta)",
                   "Power (Normal)",
                   "Sample Size",
                   "mu2",
                   "mu1")
# Display the results
Ex2

   Power (Beta) Power (Normal) Sample Size   mu2    mu1
1         0.985          0.310         100 0.012 0.0174
2         0.942          0.222         100 0.013 0.0174
3         0.879          0.150         100 0.014 0.0174
4         0.999          0.374         125 0.012 0.0174
5         0.986          0.266         125 0.013 0.0174
6         0.959          0.177         125 0.014 0.0174
7         1.000          0.435         150 0.012 0.0174
8         0.999          0.310         150 0.013 0.0174
9         0.991          0.204         150 0.014 0.0174
10        1.000          0.493         175 0.012 0.0174
11        1.000          0.353         175 0.013 0.0174
12        0.999          0.230         175 0.014 0.0174
13        1.000          0.546         200 0.012 0.0174
14        1.000          0.394         200 0.013 0.0174
15        0.999          0.257         200 0.014 0.0174
```

The results indicate small differences between the power of a two-sided t-test with equal and unequal standard deviations, while the power from `power_Beta` changes drastically without the equal precision assumption. Unlike normally distributed random variables, the beta distribution is more sensitive to the assumption of equal precision parameters. Figure 1 displays the comparison of probability density functions for beta distributed random variables with and without the equal precision assumption and the comparison for the analogous normally distributed variables with and without equal standard deviations.

Summary

This example demonstrates the use of `power_Beta` and `power_Normal`, each with equal and unequal precision parameters, to perform power analyses and sample size calculations. Since a simulation method is used within the function `power_Beta`, the computational time is dependent upon the number of trials in the simulation. It is suggested that a starting value be used, such as 100, to determine an initial range for the other parameters (e.g., range of `mu2`). Once an appropriate range of values

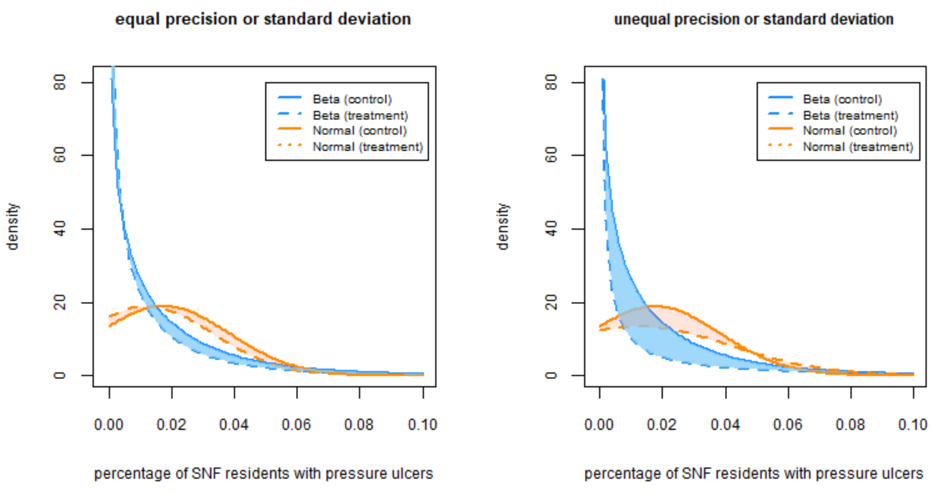


Figure 1: Comparison of equal and unequal precision or standard deviation parameters (`mu2` is assumed to be 0.0120).

is determined, the number of trials should be increased (e.g., `trials=1000`) to output more precise power and sample size estimates.

Summary and Discussion

Multiple packages are available in R to perform power analyses, including **pwr**, **MESS**, **WebPower**, and the base R **stats** package. However, these packages do not provide a comprehensive power analysis toolkit capable of calculating power or sample sizes for the test of two-sample means or ratios when the outcomes have a beta, gamma, or Poisson distribution.

The **PASSED** package extends the current power analysis functions available in R. Seven functions are provided for corresponding distributions, applying either theoretical formulas or simulation algorithms. All functions have the ability to obtain the statistical power or estimate minimum sample sizes. In particular, the formula-based approaches also support calculations for other parameters such as means and proportions. As for the simulation-based methods, users are able to customize each analysis with options to set the number of trials in the simulation and specify the assumptions for the tests. An example of how to implement and customize the functions is provided in Section Application of **PASSED**. The **PASSED** package provides a simple, one-package solution for sample size and power calculations for a wide variety of common and specialty distributions encountered in clinical research.

Computational Details

The results in this paper were obtained using R 4.0.2 and **betareg** 3.0.0. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

The authors gratefully acknowledge Dr. Richard Madsen, Professor Emeritus of Statistics University of Missouri - Columbia, for his contribution and interest in this topic. Funding for this project was provided by NIH R34AR074209A (Co-I Leary) and the Thompson Laboratory for Regenerative Orthopaedics, University of Missouri - Columbia.

Appendix

Derivation of Power Calculation Formulae for Poisson Distribution

Let x_{ij} be the number of events during the necessary study time t_i from the *jth* subject in the *ith* treatment group, $j = 1, ..., n_i$, $i = 1, 2$. It is assumed that x_{ij} are Poisson random variables with rate λ_i such that the probability function of x_{ij} is

$$P\left(x_{ij}\right) = \frac{t_i \lambda_i e^{t_i \lambda_i}}{x_{ij}!},$$

where $i = 1, 2$. Then, the total number of events in each group, denoted as X_1 and X_2, also follows a Poisson distribution:

$$X_i \sim Poisson(\lambda_i t_i n_i).$$

For the hypothesis tests:

$$H_0 : \lambda_2/\lambda_1 = R \ vs. \ H_a : \lambda_2/\lambda_1 = R' > R(one - sided)$$

and

$$H_0 : \lambda_2/\lambda_1 = R \ vs. \ H_a : \lambda_2/\lambda_1 = R' \neq R(two - sided),$$

where R denotes the pre-specified ratio of two Poisson rates. Gu et al. (2008) derives a test statistic W_5, which is asymptotically distributed as a standard normal under the null hypothesis above,

$$W_5 = \frac{2(\sqrt{X_2 + 3/8} - \sqrt{Q(X_1 + 3/8)})}{\sqrt{1+Q}},$$

where $Q = R/d$ and $d = t_1/t_2$. Then, the critical region of the one-sided test is

$$W_5 = \frac{2(\sqrt{X_2 + 3/8} - \sqrt{Q(X_1 + 3/8)})}{\sqrt{1+Q}} \geq z_{1-\alpha}. \tag{4}$$

To calculate the power under $H_a : \lambda_2/\lambda_1 = R' > R$ at significance level α, let $c = R/R'$ and multiply both sides of Equation 4 by $\sqrt{1+Q}$, which is greater than 0 as that

$$2(\sqrt{X_2 + 3/8} - \sqrt{Q(X_1 + 3/8)}) \geq z_{1-\alpha}\sqrt{1+Q}. \tag{5}$$

Add $-2(\sqrt{Q/c} - \sqrt{Q})\sqrt{X_1 + 3/8}$ to both sides of Equation 5 for the inequality,

$$2(\sqrt{X_2 + 3/8} - \sqrt{Q/c(X_1 + 3/8)}) \geq z_{1-\alpha}\sqrt{1+Q} - 2(\sqrt{Q/c} - \sqrt{Q})\sqrt{X_1 + 3/8}. \tag{6}$$

Then, divide both sides of Equation 6 by $\sqrt{1 + Q/c}$, greater than 0. It follows that

$$\frac{2(\sqrt{X_2 + 3/8} - \sqrt{Q/c(X_1 + 3/8)})}{\sqrt{1+Q/c}} \geq \frac{z_{1-\alpha}\sqrt{1+Q} - 2(\sqrt{Q/c} - \sqrt{Q})\sqrt{X_1 + 3/8}}{\sqrt{1+Q/c}}. \tag{7}$$

Under the alternative hypothesis, the left-hand side of Equation 7 is asymptotically normal distributed (Gu et al., 2008). Accordingly, the type II error, β, can be derived as:

$$\begin{aligned}
\beta &= P(H_0|H_a) \\
&= P(X < \frac{z_{1-\alpha}\sqrt{1+Q} - 2(\sqrt{Q/c} - \sqrt{Q})\sqrt{X_1 + 3/8}}{\sqrt{1+Q/c}}|H_a) \\
&= \Phi(\frac{z_{1-\alpha}\sqrt{1+Q} - 2(\sqrt{Q/c} - \sqrt{Q})\sqrt{X_1 + 3/8}}{\sqrt{1+Q/c}})
\end{aligned}$$

Incorporating $Q = R/d$, $c = R/R'$ and $X_1 = \lambda_1 t_1 n_1$, and collecting items

$$\begin{aligned}
\beta &= \Phi(\frac{z_{1-\alpha}\sqrt{1+R/d} - 2(\sqrt{(R/d)/(R/R')} - \sqrt{R/d})\sqrt{\lambda_1 t_1 n_1 + 3/8}}{\sqrt{1+(R/d)/(R/R')}}) \\
&= \Phi(\frac{z_{1-\alpha}\sqrt{\frac{R+d}{d}} - 2(\sqrt{\frac{R'}{d}} - \sqrt{\frac{R}{d}})\sqrt{\lambda_1 t_1 n_1 + 3/8}}{\sqrt{\frac{R'+d}{d}}}) \\
&= \Phi(\frac{z_{1-\alpha}\sqrt{\frac{R+d}{R'}} - 2(1 - \sqrt{\frac{R}{R'}})\sqrt{\lambda_1 t_1 n_1 + 3/8}}{\sqrt{\frac{R'+d}{R'}}}) \\
&= \Phi(\frac{z_{1-\alpha}C - A\sqrt{B}}{D}),
\end{aligned}$$

where $A = 2(1 - \sqrt{\frac{R}{R'}})$,
$B = \lambda_1 t_1 n_1 + 3/8$,
$C = \sqrt{\frac{R+d}{R'}}$,
$D = \sqrt{\frac{R'+d}{R'}}$
So, power can be expressed as

$$\begin{aligned}
Power(W_5) &= 1 - \Phi(\frac{z_{1-\alpha}C - A\sqrt{B}}{D}) \\
&= \Phi(\frac{A\sqrt{B} - z_{1-\alpha}C}{D}).
\end{aligned} \tag{8}$$

Moreover, using $z_{power} = \Phi^{-1}(Power)$, Equation 8 can be expressed as:

$$z_{power} = \frac{A\sqrt{B} - z_{1-\alpha}C}{D}. \tag{9}$$

Solving Equation 9 for B,

$$B = (\frac{z_{power}D - z_{1-\alpha}C}{A})^2. \tag{10}$$

Since $B = \lambda_1 t_1 n_1 + 3/8$, the sample size calculation formula of one-sided test can be determined by solving Equation 10 for n_1

$$n_1 = \frac{(\frac{z_{1-\alpha}C + z_{power}D}{A})^2 - \frac{3}{8}}{\lambda_1 t_1} (one - sided)$$

and the two-sided test can be derived similarly as

$$n_1 = \frac{(\frac{z_{1-\frac{\alpha}{2}}C + z_{power}D}{A})^2 - \frac{3}{8}}{\lambda_1 t_1} (two - sided).$$

Bibliography

C. L. Aberson. *Applied power analysis for the behavioral sciences*. Routledge, 2019. URL https://doi.org/10.4324/9781315171500. [p542]

E. Brittain and J. J. Schlesselman. Optimal allocation for the comparison of proportions. *Biometrics*, pages 1003–1009, 1982. URL https://doi.org/10.2307/2529880. [p543]

G. Casella and R. L. Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002. ISBN 978-0-357-75313-2. [p550]

S. Champely, C. Ekstrom, P. Dalgaard, J. Gill, S. Weibelzahl, A. Anandkumar, C. Ford, R. Volcic, and H. De Rosario. pwr: Basic functions for power analysis, 2017. URL https://cran.r-project.org/web/packages/pwr/index.html. [p542, 548]

C.-H. Chang, J.-J. Lin, and N. Pal. Testing the equality of several gamma means: a parametric bootstrap method with applications. *Computational Statistics*, 26(1):55–76, 2011. URL https://doi.org/10.1007/s00180-010-0209-1. [p551]

M. R. Chernick and C. Y. Liu. The saw-toothed behavior of power versus sample size and software solutions: Single binomial proportion using exact methods. *The American Statistician*, 56(2):149–155, 2002. URL https://doi.org/10.1198/000313002317572835. [p550]

S.-C. Chow, H. Wang, and J. Shao. *Sample size calculations in clinical research*. Chapman and Hall/CRC, 2007. URL https://doi.org/10.1002/wics.155. [p542]

N. Cressie and H. Whitford. How to use the two sample t-test. *Biometrical Journal*, 28(2):131–148, 1986. URL https://doi.org/10.1002/bimj.4710280202. [p547]

F. Cribari-Neto and A. Zeileis. Beta regression in R. *Journal of Statistical Software*, 34(2):1–24, 2010. URL https://doi.org/10.18637/jss.v034.i02. [p549]

M. D. Huffman. An improved approximate two-sample poisson test. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 33(2):224–226, 1984. URL https://doi.org/10.2307/2347448. [p546]

C. T. Ekstrøm. *The R primer*. CRC Press USA, 2012. URL https://doi.org/10.1201/9781315154411. [p542, 548]

S. Ferrari and F. Cribari-Neto. Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7):799–815, 2004. URL https://doi.org/10.1080/0266476042000214501. [p549]

J. L. Fleiss, A. Tytun, and H. K. Ury. A simple approximation for calculating sample sizes for comparing independent proportions. *Biometrics*, pages 343–346, 1980. URL https://doi.org/10.2307/2529990. [p543]

S. Gates, I. Nguyen, M. Del Core, P. A. Nakonezny, H. Bradley, and M. Khazzam. Incidence and predictors of positive intraoperative cultures in primary shoulder arthroplasty following prior ipsilateral shoulder surgery. *JSES International*, 2020. URL https://doi.org/10.1016/j.jseint.2019.12.011. [p544]

K. Gu, H. K. T. Ng, M. L. Tang, and W. R. Schucany. Testing the ratio of two poisson rates. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 50(2):283–298, 2008. URL https://doi.org/10.1002/bimj.200710403. [p546, 547, 556]

A. K. Gupta and S. Nadarajah. *Handbook of Beta Distribution and Its Applications*. CRC Press, New York, 2004. URL https://doi.org/10.1201/9781482276596. [p548]

J. M. Hilbe. *Negative binomial regression*. Cambridge University Press, 2011. ISBN 978-0-521-19815-8. [p544]

Y.-R. Hong, R. G. Salloum, S. Yadav, G. Smith, and A. G. Mainous III. Patient–provider discussion about cancer treatment costs and out-of-pocket spending: Implications for shared decision making in cancer care. *Value in Health*, 2020. URL https://doi.org/10.1016/j.jval.2020.08.002. [p550]

C. Hu, H. Zhou, and A. Sharma. Application of beta-distribution and combined uniform and binomial methods in longitudinal modeling of bounded outcome score data. *The AAPS Journal*, 22, 2020. URL https://doi.org/10.1208/s12248-020-00478-5. [p548]

M. K. Hutchinson and M. C. Holtman. Analysis of count data using poisson regression. *Research in nursing & health*, 28(5):408–418, 2005. URL https://doi.org/10.1002/nur.20093. [p546]

S.-L. Jan and G. Shieh. Optimal sample sizes for welch's test under various allocation and cost considerations. *Behavior research methods*, 43(4):1014–1022, 2011. URL https://doi.org/10.3758/s13428-011-0095-7. [p548]

S. Jones, S. Carley, and M. Harrison. An introduction to power and sample size estimation. *Emergency Medicine Journal*, 20(5):453–458, 2003. URL https://doi.org/10.1136/emj.20.5.453. [p542]

M. Kohl. *MKmisc: Miscellaneous functions from M. Kohl*, 2021. URL https://www.stamats.de. R package version 1.8. [p542]

B. LeBeau. *Power Analysis by Simulation using R and simglm*, 2019. URL https://doi.org/10.17077/f7kk-6w7f. [p542]

C. B. G. LEITE, L. V. Ranzoni, P. N. Giglio, M. B. Bonadio, L. D. P. Melo, M. K. Demange, and R. G. Gobbi. Assessment of the use of tranexamic acid after total knee arthroplasty. *Acta Ortopédica Brasileira*, 28(2):74–77, 2020. URL https://doi.org/10.1590/1413-785220202802228410. [p543]

C. Luan, D.-T. Xu, N.-J. Chen, F.-F. Wang, K.-S. Tian, C. Wei, and X.-B. Wang. How to choose kinematic or mechanical alignment individually according to preoperative characteristics of patients? *BMC Musculoskeletal Disorders*, 21(1):1–6, 2020. URL https://doi.org/10.1186/s12891-020-03472-2. [p547]

K. Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50 (302):157–175, 1900. URL https://doi.org/10.1080/14786440009463897. [p543]

D. Plessl, B. Salomon, A. Haydel, C. Leonardi, A. Bronstone, and V. Dasa. Rapid versus standard recovery protocol is associated with improved recovery of range of motion 12 weeks after total knee arthroplasty. *The Journal of the American Academy of Orthopaedic Surgeons*, 2020. URL https://doi.org/10.5435/JAAOS-D-19-00597. [p542, 546]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. ISBN 3-900051-07-0. [p542, 550]

Scherer. *Package 'samplesize'*, 2016. URL https://cran.r-project.org/web/packages/samplesize/index.html. [p542]

P. T. Schickedanz and G. F. Krause. A test for the scale parameters of two gamma distributions using the generalized likelihood ratio. *Journal of applied meteorology*, 9(1):13–16, 1970. URL https://doi.org/10.1175/1520-0450(1970)009<0013:ATFTSP>2.0.CO;2. [p552]

W.-K. Shiue and L. J. Bain. Experiment size and power comparisons for two-sample poisson tests. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(2):130–134, 1982. URL https://doi.org/10.2307/2347975. [p546]

W.-K. Shiue and L. J. Bain. A two-sample test of equal gamma distribution scale parameters with unknown common shape parameter. *Technometrics*, 25(4):377–381, 1983. URL https://doi.org/10.1080/00401706.1983.10487901. [p551]

W.-K. Shiue, L. J. Bain, and M. Engelhardt. Test of equal gamma-distribution means with unknown and unequal shape parameters. *Technometrics*, 30(2):169–174, 1988. URL https://doi.org/10.1080/00401706.1988.10488364. [p551]

M. Smithson and J. Verkuilen. A better lemon squeezer? maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, 11(1):54–71, 2006. URL https://doi.org/10.1037/1082-989X.11.1.54. [p549]

H. C. Thode. Power and sample size requirements for tests of differences between two poisson rates. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 46(2):227–230, 1997. URL https://doi.org/10.1111/1467-9884.00078. [p546]

E. Zhang, V. Q. Wu, S.-C. Chow, and H. G. Zhang. *Package 'TrialSize'*, 2013. URL https://cran.r-project.org/web/packages/TrialSize/index.html. [p542]

Z. Zhang and Y. Mai. *WebPower: Basic and Advanced Statistical Power Analysis*, 2018. URL https://CRAN.R-project.org/package=WebPower. R package version 0.5.2. [p542]

H. Zhu and H. Lakkis. Sample size calculation for comparing two negative binomial rates. *Statistics in medicine*, 33(3):376–387, 2014. URL https://doi.org/10.1002/sim.5947. [p544, 545]

Jinpu Li
University of Missouri
USA
E-mail: lijinp@health.missouri.edu

Ryan P. Knigge
University of Minnesota
USA
E-mail: knigg008@umn.edu

Kaiyi Chen
University of Missouri
USA
E-mail: kcvpc@mail.missouri.edu

Emily V. Leary, Ph.D.
Thompson Laboratory for Regenerative Orthopaedics
Department of Orthopaedic Surgery
University of Missouri
USA
E-mail: learye@health.missouri.edu

spNetwork: A Package for Network Kernel Density Estimation

by Jeremy Gelb

Abstract This paper introduces the new package **spNetwork** that provides functions to perform Network Kernel Density Estimate analysis (NKDE). This method is an extension of the classical Kernel Density Estimate (KDE), a non parametric approach to estimate the intensity of a spatial process. More specifically, it adapts the KDE for cases when the study area is a network, constraining the location of events (such as accidents on roads, leaks in pipes, fish in rivers, etc.). We present and discuss in this paper the three main versions of NKDE: simple, discontinuous, and continuous that are implemented in **spNetwork**. We illustrate how to apply the three methods and map their results using a sample from a real dataset representing bike accidents in a central neighborhood of Montreal. We also describe the optimization techniques used to reduce calculation time and investigate their impacts when applying the three NKDE to a city-wide dataset.

Introduction

Data representing events in space are collected in many research fields. The analysis of these points clouds is crucial in identifying hot spots, e.g., complaints about environmental noise, cases of diseases, observation of animals or plant species, crimes, etc. The Kernel Density Estimate (KDE) is a nonparametric method often used to estimate the intensity function of a spatial process from a sample of events. More specifically, the KDE is part of the point pattern analysis (PPA) family of methods. It allows for the analysis of first-order properties (variation of density) of a spatial process (Baddeley et al., 2015). Recent works (Xie and Yan, 2008; Okabe et al., 2009; McSwiggan et al., 2017) have shown that the KDE method is unadapted when events are constrained on a network (road crashes, leaks in a network of pipes, crimes reported in streets, etc.). The classical KDE is based on the hypothesis of an infinite, homogenous, two-dimensional space, which is a very rough approximation if the study area is a network. Indeed, a network is a particular space between a simple line (1D) and a plan (2D). In a network, the movement is constrained on multiple one-dimensional lines, event if it is possible to change direction at intersections. Steenberghen et al. (2010) call it a 1.5D space. Moreover, the reticular distance between objects on a network is always superior or equal to the Euclidean distance. Thus, using the latter leads to underestimations of the real distance between the objects on a network. The need to extend the KDE method is part of a more general trend started in the 1990's aiming to generalize the spatial analysis to network spaces (Okabe and Satoh, 2006; Xie and Yan, 2008). At first, several adaptations were proposed but with a limited relevance because of strong limitations (applied to very simple networks or a simple spatial aggregation of a classical KDE) (Flahaut et al., 2003; Borruso, 2003; Porta et al., 2009). More recent works have developed methods overcoming these first limitations, forming the set of the NKDE methods.

Currently, these methods are not easily accessible. Xie and Yan (2008) developed an ArcMap (ESRI, 2017) plugin proposing only a biased density estimator; Hwang and Winslow (2012) developed GeodaNet (GeodaCenter, university of Chicago), but it is no longer accessible on their website; Okabe et al. (2006) proposed SANET, a toolbox for network analysis as an ArcMap plugin or a standalone application. Even though SANET is a free software, it is not open-source, and the actual license limits the use to research only. Moreover, a new user must send a request to the maintainer to obtain an activation key. Finally, the R package **spatstat** (Baddeley and Turner, 2005), dedicated to PPA (Baddeley et al., 2004), includes a function to perform NKDE analysis (`density.lpp`), but it returns only rasters and requires an extremely long calculation time for large and medium-sized datasets. Thus, accessible tools to perform NKDE analysis are missing. This encourages researchers and professionals to use classical KDE in cases where NKDE is much more adapted and limits the reproducibility of research.

We propose the **spNetwork** package to fill this gap. It implements three estimators of the intensity of a spatial process on a network, the possibility to use adaptive bandwidths, and several optimization methods to ensure reasonable calculation time. R provides a perfect environment to implement the NKDE method, considering its growing community, its capacity to read, write and manipulate geographical data (**rgdal**, **sp**, **rgeos**, **maptools**, (Bivand et al., 2020; Pebesma and Bivand, 2020; Bivand and Rundel, 2020; Bivand and Lewin-Koh, 2020)), the existence of libraries to manipulate networks (**igraph** (Csardi and Nepusz, 2020)), and the facilitated interface to C++ with **Rcpp** (Eddelbuettel et al., 2020a).

In the first section of this paper, we present the classical KDE and introduce the main concepts of the method. In the second section, we introduce the three NKDE estimators and briefly compare their respective advantages and limits. The three NKDE are then applied to a small dataset about road accidents involving a cyclist in a central neighborhood of Montreal. Finally, we investigate the calculation time of the three NKDE with different settings considering that calculation time is an important issue for the NKDE method.

Kernel density estimate

General description

For a spatial process p, represented by a set of events e, its intensity function λ at location u ($\lambda(u)$) can be estimated in a non parametric way by kernel estimation (Silverman, 1986). Typically, in a two-dimensional space, a regular grid is defined on the study area, and the intensity is estimated at the centers of each quadra (pixels). At each location, u, the events within a specified bandwidth, bw, contributes to the local estimated intensity. The strength of this contribution depends on the distance between the events and u, the event's weight, and the kernel function. This function distributes the mass of the events within a circular area around each event. The radius of this area is called the bandwidth (or the standard deviation of the kernel). A larger bandwidth produces smoother results and higher bias but reduces variance. The Kernel Density Estimate can thus be obtained as follows:

$$\lambda(u) = \frac{1}{bw^2} \sum_{i=1}^{n} w_i \cdot K(dist(u, e_i)), \tag{1}$$

with n, the number of events that satisfy $dist(u, e_i) < bw$, w_i the weight of the event e_i and K the kernel function. K must be a probability density function and verifies the two following conditions:

$$\begin{aligned} &K(x) > 0 \text{ if } x < bw \\ &K(x) = 0 \text{ if } x \geq bw \\ &\int_{-\infty}^{+\infty} K(x) = 1. \end{aligned} \tag{2}$$

Many kernel functions exist; Figure 1 shows the most commonly used (and implemented in **spNetwork**).

```
library(ggplot2)
library(tidyr)
library(spNetwork)
library(RColorBrewer)
library(kableExtra)
library(dplyr)

x <- seq(-15.01, 15.01, by = 0.01)
kernels_func <- list(gaussian_kernel, epanechnikov_kernel, quartic_kernel,
                     triangle_kernel, tricube_kernel, triweight_kernel,
                     cosine_kernel, uniform_kernel)

cols <- lapply(kernels_func, function(f){f(x, 15)})
df <- data.frame(do.call(cbind, cols))
names(df) <- c("Gaussian", "Epanechnikov", "Quartic", "Triangle",
               "Tricube", "Triweight", "Cosine", "Uniform")
df$x <- x

pivot_cols <- names(df)[names(df)!="x"]
df2 <- pivot_longer(df,cols = pivot_cols)
names(df2) <- c("x","kernel","y")

ggplot(df2) +
  geom_line(aes(x = x, y = y, color = kernel), size = 1)+
  xlim(c(-15.01, 15.01))+
  scale_color_brewer(palette = "Accent")+
  scale_y_continuous(name = "density")
```

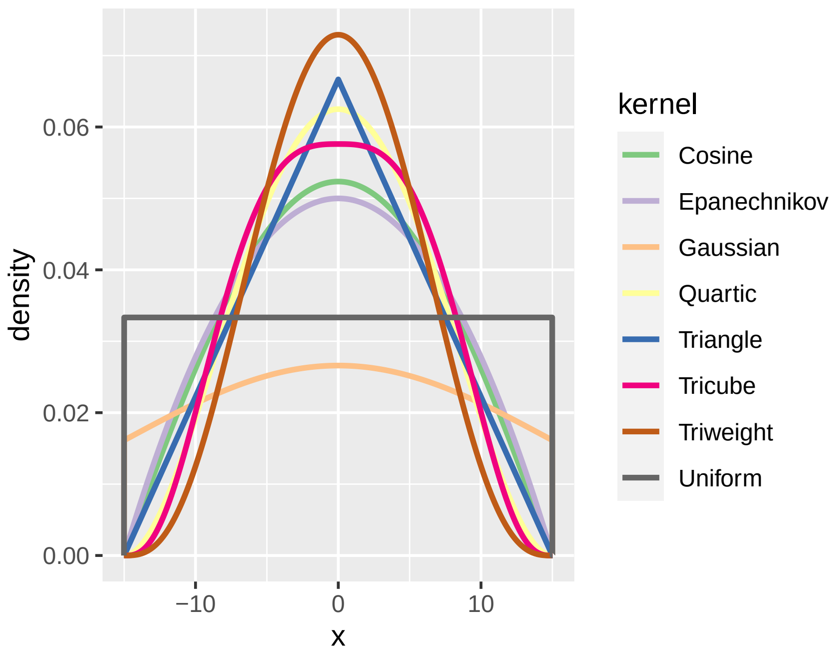


Figure 1: The different kernels implemented in **spNetwork**.

Many authors have stressed that the choice of the Kernel function has a smaller impact on final results in comparison with the choice of the bandwidth (O'Sullivan and Wong, 2007; Xie and Yan, 2008; Turlach, 1993). The Gaussian kernel is a special case (non-compact kernel) because it does not integrate to one on the domain $[-bw; +bw]$ but on $[-\infty; +\infty]$, leading to a loss of mass for each event.

Diggle's correction

Formula 1 describes the basic KDE value, which is unbiased only if the study area is infinite in every direction and if the spatial process is sampled everywhere in this space. In practice, such situations are rare, and the basic KDE is biased at the frontier of the study area. For example, let us imagine a case where abandoned syringes locations are systematically reported in a specific district of a city. The ones lying outside the district limits are not considered, leading to a situation where the border areas systematically have a lower estimated intensity. To reduce this effect, Diggle's correction (equation 3) is generally used (Diggle, 1985).

$$\begin{gathered}\lambda^D(u) = \frac{1}{bw^2}\sum_{i=1}^{n} w_i \cdot \frac{1}{e(e_i)}K(dist(u,e_i)) \\ e(u) = \int_W^v K(dist(u,v)).\end{gathered} \tag{3}$$

This correction increases the weight of the events located close to the study area border because a part of their mass is lost above the border. For an event e_i, with the location u, $e(u)$ is the fraction of its mass in the study area (W). The correction factor is thus inversely proportional to the fraction of the event's mass in the study area. For example, an event having only half its mass in the study area will end up with a doubled weight.

Adaptive bandwidth

In its basic form, the bandwidth of the KDE is fixed, i.e. the value of the bandwidth is the same everywhere in the study area. This is an important limitation if the intensity of the spatial process has a pronounced spatial variation. In that case, the risk is to obtain oversmoothed results in areas with high intensity and undersmoothed results in areas with low intensity. To overcome this limitation, it is possible to replace the fixed bandwidth with a vector of spatially varying bandwidths. The bandwidths should be smaller in subregions with higher intensity and wider elsewhere. The Abramson method (Abramson, 1982) is often used to define this vector of bandwidths. Its calculation requires two steps. First, a pilot estimate ($\tilde{\lambda}$) is calculated at the location of each event (u_{ei}) by using a global bandwidth h_0. Second, the vector of bandwidths ($h(u_{ei})$) is calculated with equation 4:

$$h(u_{ei}) = h_0 \cdot \frac{1}{\sqrt{\tilde{f}(u_{ei})/\gamma}} \qquad (4)$$
$$\gamma = (\prod_{i=1}^{n} \frac{1}{\sqrt{\tilde{f}(e_i)}})^{\frac{1}{n}}.$$

A trimming value can be defined to avoid obtaining large bandwidths in subregions with low intensity. Let us recall here that the adaptive bandwidth is still based on the choice of an arbitrary global bandwidth and is not a bandwidth selection method.

The three Network Kernel Density Estimate

We present the three NKDE methods implemented in the **spNetwork** package in this section .

The simple NKDE

A first NKDE method was proposed by Xie and Yan (2008) and constituted a geographical attempt to extend the planar KDE to the network case. It received some attention in geography (Xie and Yan, 2013) but has been criticized for its statistical incorrectness (detailed later). We refer to this method as the "*simple NKDE*", and it can be summarized by three points:

- The intensity of the spatial process is only estimated on the network. Its edges are split into lixels (one-dimensional pixels), and the centers of the lixels are used as locations for estimating intensity.
- The distances between events and sampling points are calculated as the shortest path distances on the network instead of Euclidean distances.
- The intensity function is slightly modified (equation 5).

$$\lambda(u) = \frac{1}{bw} \sum_{i=1}^{n} K(dist(u, e_i)). \qquad (5)$$

The method is appealing for three reasons. First, from a purely geographical point of view, the extension of the planar KDE to network KDE is intuitive. The modification applied is in line with other methods in PPA extended from planar to network cases like the K-function or the nearest neighbor analysis (Okabe and Sugihara, 2012). Second, the method does not rely on an expensive algorithm and thus achieves a short calculation time. Third, the adapted formula makes the interpretation straightforward: it "estimates the density over a linear unit" rather than an area unit (Xie and Yan, 2008, pp. 398). Note that the modification of the equation can also be applied to the next methods.

To get a visual representation of the method, we consider the situation depicted in Figure 2. The lines constitute the network, and a red dot is a single event.

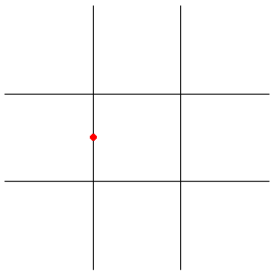


Figure 2: A single event on a network.

We can evaluate the density of the spatial process with the simple NKDE and visualize the result in 3D (Figure 3), using density as the height.

At intersections on the network, the mass of the event is multiplied in each direction. Consequently, the simple NKDE is not a real kernel density function because it does not integrate to 1 on its domain.

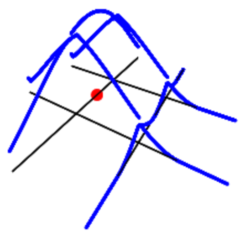


Figure 3: 3D visualization of the simple NKDE.

The practical consequence is the systematic overestimation of the density, which could be problematic in subregions with many events. To overcome this limitation, Okabe et al. (2009) proposed two unbiased estimators: the discontinuous NKDE and the continuous NKDE.

The discontinuous NKDE

Considering the problem inherent to the simple NKDE, the discontinuous NKDE proposes a simple solution. The mass is divided at intersections according to the number of directions minus one. This is easily represented by Figure 4.

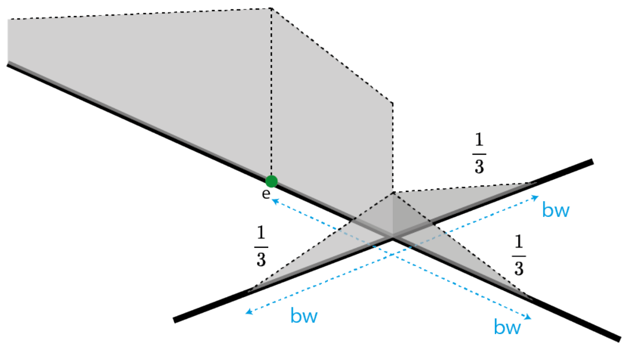


Figure 4: Graphical depiction of the discontinuous NKDE.

With the same example as previous, we obtain Figure 5.

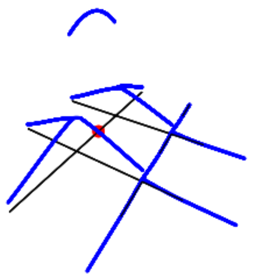


Figure 5: 3D visualization of the discontinuous NKDE.

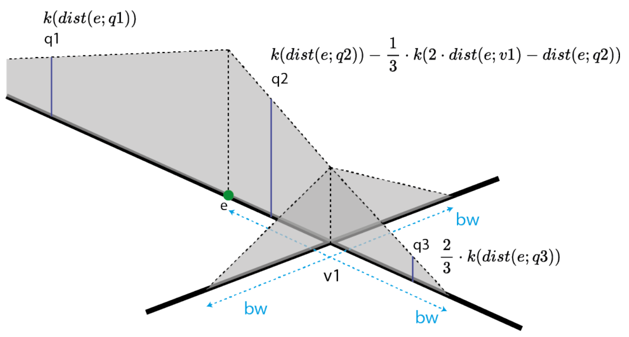
This estimator is unbiased. However, its discontinuous nature can be counter-intuitive in real application. For example, considering the analysis of crimes on a network, it would be counter-intuitive that the “influence” of a crime suddenly drops from one street to another. The discontinuous NKDE can be summarized with Equation 6.

$$K(dist(u,e_i)) = \frac{2k(dist(u,e_i))}{n_{i1}\prod_{i=i}^{j}(n_{ij}-1)}, \tag{6}$$

with k the kernel function, n_{ij} the number of edges connected at the intersection j on the path originating at the event, e_i, and ending at the location, u.

The continuous NKDE

Finally, the continuous NKDE attempts to combine the best of the two worlds: it adjusts the values of the NKDE at intersections to ensure that it integrates to 1 on its domain and applies a backward correction to force the density values to be continuous. A simple case is represented in Figure 6.



$k(dist(e;q1))$
q1
$k(dist(e;q2)) - \frac{1}{3} \cdot k(2 \cdot dist(e;v1) - dist(e;q2))$
q2
e
bw
q3
$\frac{2}{3} \cdot k(dist(e;q3))$
v1
bw
bw

Figure 6: Graphical depiction of the continuous NKDE.

In this simple case, there are three different equations to calculate the kernel density (here, q1, q2, q3). Considering the previous simple example, we obtain Figure 7.

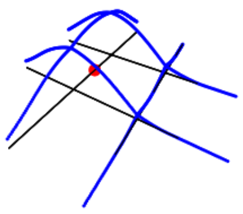


Figure 7: 3D visualization of the continuous NKDE.

Because of its backward correction, the continuous NKDE is recursive in nature and can hardly be presented as an equation. Okabe and Sugihara (2012) describe this method with a recursive function, implemented in **spNetwork**. Indeed, for each node encountered, a correction factor must be applied to all the previous edges traveled within the remaining bandwidth, and this correction must also be split between all the nodes encountered in that direction. In comparison, for the two previous methods, the algorithm only has to flow in one direction along the edges from the event. As a consequence, much more iterations are required for the continuous NKDE. More specifically, Okabe et al. (2009) state that the complexity of the discontinuous (and by extension, of the simple) NKDE algorithm is $O(nn_L)$ with n_L the number of edges and n the number of events. For the continuous NKDE, the complexity of the algorithm is a function of the ratio of the bandwidth and the length of the edges. For this algorithm, Okabe et al. (2009) observed that the computation time has the following lower limit (equation 7):

$$\prod_{i=1}^{i_{max}} (bw - d)/(d_{i+1} - d_i), \tag{7}$$

with d_i the distance from an event to the ith-nearest node, and i_{max} the last node verifying $d_i < bw$. In other words, having many short edges in a network or using a larger bandwidth will increase the calculation time exponentially for the continuous NKDE.

The pros and cons of the three methods are summarized in Table 1.

Method	Pros	Cons
Simple NKDE	Continuous / Easy calculation	Biased, not a true kernel
Discontinuous NKDE	Unbiased / Easy calculation	Discontinuous
Continuous NKDE	Unbiased / Continuous	Time consuming

Table 1: Pros and cons of the three NKDE.

The two KDE extensions presented in section 2 (adaptive bandwidth and edge correction) can be directly transposed to these three NKDE methods.

Implementation in spNetwork

The **spNetwork** package contains two "high- level functions" and three "helper functions". The "high level functions" `nkde` and `nkde.mc` can be used to calculate the three NKDE presented in the previous section. Both functions use exactly the same parameters, `nkde.mc` is a version of `nkde` allowing for multiprocessing via the package **future** (Bengtsson, 2020a). The three main inputs are:

- `lines`, a "SpatialLinesDataFrame" (object defined in the **sp** package), representing the lines of the network.
- `events`, a "SpatialPointsDataframe" (object defined in the **sp** package), representing the realizations of the spatial process.
- `samples`, a "SpatialPointsDataframe" providing the locations where the density must be estimated.

These three inputs must only have simple and valid geometries and the same coordinates reference system. Otherwise, an error is raised by the function before starting any calculation. The parameters `method`, `kernel_name` and `bw`, indicate respectively which NKDE (simple, discontinuous, or continuous), and which kernel function (gaussian, quartic, triweight, etc.) to use, and the bandwidth of the kernel. `adaptive` and `diggle_correction` allow the user to specify if an adaptive bandwidth and Diggle's correction must be calculated. Finally, the remaining parameters control the geometric precision of the network and the optimization aspects.

The three "helper functions" are provided to facilitate the creation of the sampling points on the network:

- The function `lixelize_lines` splits the lines of a "SpatialLinesDataFrame" according to a selected distance to generate lixels.
- The function `lines_center` returns for each line of a "SpatialLinesDataFrame" its center points (the point located at mid-distance between the start and the end of the linestring).
- The function `lines_points_along` returns points located along the lines of a "SpatialLines-DataFrame", evenly spaced according to a selected distance.

Considering the previous functions, the workflow of NKDE analysis with **spNetwork** follows five steps:

1. Loading the network dataset and the event dataset as sp objects, typically with the function `readOGR` from the package **rgdal**. Many standard formats can thus be used such as ESRI Shapefile, Geopackage, GeoJson, SpatialLite, etc.
2. Generating the sampling points on the network by using the helper functions.
3. Calculating the density estimations with the `nkde` or `nkde.mc` functions.
4. Adding the densities as a new column to the sampling "SpatialPointsDataFrame" or lixels "SpatialLinesDataFrame".
5. Exporting the results, typically with `writeOGR` from **rgdal** to map the results with a dedicated mapping software (like Qgis (QGIS Development Team, 2020)).

Optimization

The main problem with NKDE methods is the calculation time. Building the network and calculating the paths between its nodes can be a costly process (depending on the size of the network studied and the number of events). The implementation in **SpNetwork** uses many techniques to reduce the calculation time.

First, only the events are added as nodes in the network. The sampling points are snapped to their nearest edge. Each edge in the network is a straight line. Thus, the final distance between a sampling point and a node of the network is calculated with the Euclidean distance. Consequently, the complexity of the network is not affected by the density of the sampling points.

Second, the calculus of the density is event-oriented. In other words, the kernel density is sequentially calculated around each event. The density values of the samples are updated at each iteration. The advantage of this approach is that the calculation time is less affected by the density of the sampling points.

Third, the events can be aggregated, and their weights added within a threshold distance. In many applications, a small distance between events is not significant considering the accuracy of the location method (geocoding, smartphone GPS, location at intersections, etc.). Aggregating events can simplify networks and limit the number of iterations when calculating the NKDE.

Fourth, following the idiom "divide and conquer", the user can specify the shape of a grid on the study area to split the calculation. In that case, for each quadra of the grid, the sampling points in the quadra are selected. The events and the lines of the network are selected if they intersect a buffer around the quadra (the size of the buffer being the bandwidth of the kernel). Thus, the separate calculations do not produce edge effects. All the spatial queries are optimized using the quadtree spatial index proposed by the package **SearchTrees** (Becker, 2012). This approach limits the risk of memory issues and increases calculation speed for big datasets.

Fifth, when the dataset is split, the calculation can be separated between several processes. Considering the fact that a lot of computers are now equipped with processors having more than four cores, dividing the work between them is an easy way to reduce the overall calculation time. The packages **future** and **future.apply** (Bengtsson, 2020b) are used in **spNetwork** to support multiprocessing. It ensures good compatibility between OS and even has features to dispatch calculations on multiple computers.

Finally, the main algorithms are implemented with **Rcpp** and **RcppArmadillo** (Eddelbuettel et al., 2020b) to overcome the R (scripted language) limitations when loops and recursions are involved (Ligges and Fox, 2008) and benefit from faster C++ compiled code.

Example with bike accidents in Montreal

In this section, we illustrate the use of the package with a built-in dataset. This dataset is an extract from the Montreal Open Data website representing bike accidents that occurred on the Montreal road network in 2017 (Figure 8).

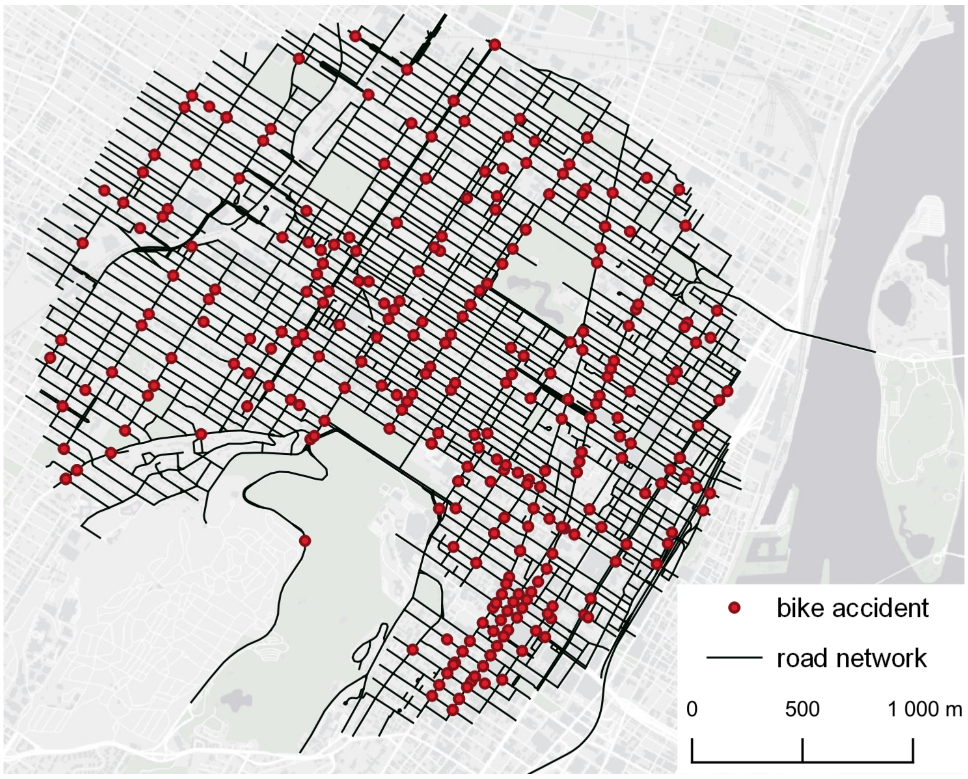


Figure 8: Example dataset of bike accidents in Montreal.

To identify hot spots of bike accidents, we calculated the three NKDE (simple, discontinuous, and continuous) with a quartic kernel (Figure 9). We selected a 200 meters bandwidth considering that the mean length of a road segment in this network is 108 meters. This bandwidth ensures that, most often, only events located closer than two street segments from a sampling point contribute to its density.

```
library(spNetwork)
library(rgdal)

## Step 1: Loading the data
networkgpkg <- system.file("extdata", "networks.gpkg",
                           package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg",
                          package = "spNetwork", mustWork = TRUE)
mtl_network <- readOGR(networkgpkg,layer="mtl_network",verbose = FALSE)
bike_accidents <- readOGR(eventsgpkg,layer="bike_accidents", verbose = FALSE)

## Step 2: Generating the sampling points on the network
# splitting the lines as lixels
lixels <- lixelize_lines(mtl_network, 100, 50)

# extracting the center of lixels as sampling points
sample_pts <- lines_center(lixels)

## Step 3: Calculating the densities estimations
# densities for the simple NKDE
nkde_simple <- nkde(lines = mtl_network,
                         events = bike_accidents,
                         w = rep(1, nrow(bike_accidents)),
                         samples = sample_pts,
                         kernel_name = "quartic",
                         bw = 200, method = "simple",
                         div = "bw", digits = 2, tol = 0.01,
                         agg = 10, grid_shape = c(1, 1),
                         verbose = FALSE)

# densities for the discontinuous NKDE
nkde_discontinuous <- nkde(lines = mtl_network,
                                events = bike_accidents,
                                w = rep(1, nrow(bike_accidents)),
                                samples = sample_pts,
                                kernel_name = "quartic",
                                bw = 200,method = "discontinuous",
                                div = "bw", digits = 2, tol = 0.01,
                                agg = 10, grid_shape = c(1, 1),
                                verbose = FALSE)

# densities for the continuous NKDE
nkde_continuous <- nkde(lines = mtl_network,
                                events = bike_accidents,
                                w = rep(1, nrow(bike_accidents)),
                                samples = sample_pts,
                                kernel_name = "quartic",
                                bw = 200, method = "continuous",
                                div = "bw",digits = 2, tol = 0.01,
                                agg = 10, grid_shape = c(1, 1),
                                verbose = FALSE)
```

To obtain more readable results, one can multiply the obtained densities by the total number of accidents (to make the spatial integral equal to the number of events) and multiply this value again by 1000 to get the estimated numbers of accidents per kilometer.

```
## Step 4: Adding the densities as new columns to the sampling
lixels$simple_density <- nkde_simple * nrow(bike_accidents) * 1000
lixels$continuous_density <- nkde_continuous * nrow(bike_accidents) * 1000
lixels$dicontinuous_density <- nkde_discontinuous * nrow(bike_accidents) * 1000
lixels$difference <- lixels$simple_density - lixels$continuous_density
```

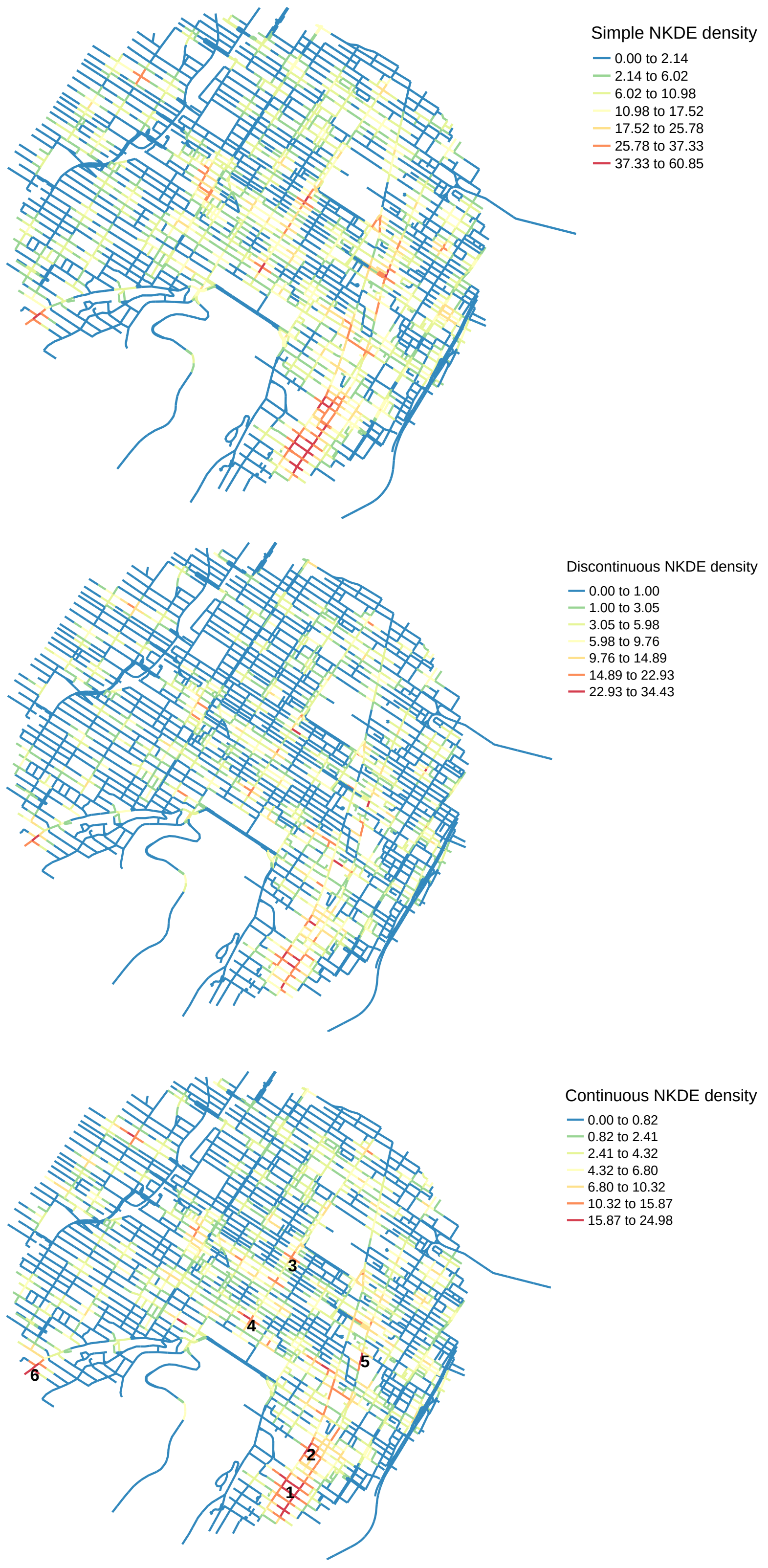


Figure 9: Estimated densities of the three NKDE for bike accidents in Montreal.

As one can observe in Figure 9, the three methods highlight the same major hot spots. We can identify a primary area in the downtown with two distinct clusters (1 and 2). Then, more local hot spots are also visible on the three maps (3, 4, 5, 6). All of them occur at intersections on streets with the main bicycle paths.

Figure 10 shows the values obtained for each sampling point for the three NKDE, sorted with the value of the discontinuous NKDE. We can observe that the simple NKDE tends to produce higher values and thus seriously overestimate the density of accidents in hot spot regions.

```
library(ggplot2)
library(tidyr)

df <- lixels@data[c("simple_density",
                    "continuous_density",
                    "dicontinuous_density")]

df <- df[order(df$dicontinuous_density), ]
df$oid <- 1:nrow(df)

pivot_cols <- names(df)[names(df) != "oid"]
df2 <- pivot_longer(df, cols = pivot_cols)
df2$name <- case_when(
  df2$name == "simple_density" ~ "Simple NKDE",
  df2$name == "continuous_density" ~ "Continuous NKDE",
  df2$name == "dicontinuous_density" ~ "Discontinuous NKDE")

ggplot(df2) +
  geom_point(aes(x = oid, y = value, color = name), size = 0.2)+
  ylab("Estimated densities")+
  scale_color_manual(values = c("Simple NKDE"="#F8766D",
                                "Continuous NKDE"="#00BA38",
                                "Discontinuous NKDE"="#619CFF"))+
  theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
        axis.ticks.x = element_blank(), legend.title = element_blank())
```

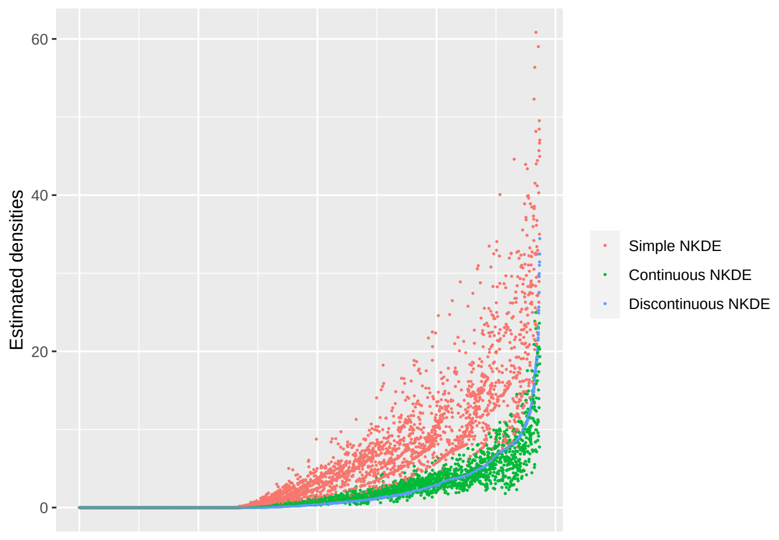


Figure 10: Comparison of the estimated densities.

Adaptive bandwidth

An adaptive bandwidth based on the Abramson's inverse-square-root rule (Abramson, 1982) can be used to produce a new version of the three NKDE. We will only calculate here the results for the continuous NKDE. Figure 11 shows the new results and the size of some calculated bandwidths with circles (not all the bandwidths are shown; otherwise the map would not be readable). As we can see, the adaptive kernel produces more smoothed results in low-intensity subregions and more detailed results in high-density regions. We can see now that the primary hot spot (1) seems concentrated on a smaller area.

```
library(rgeos)

# calculating the continuous kernel with an adaptive bandwidth
nkde_abw <- nkde(lines = mtl_network,
                             events = bike_accidents,
                             w = rep(1, nrow(bike_accidents)),
                             samples = sample_pts,
                             kernel_name = "quartic",
                             bw = 200,
                             adaptive = TRUE, trim_bw = 400,
                             method = "continuous",
                             div = "bw",
                             digits = 2, tol = 0.01,
                             agg = 10, grid_shape = c(1, 1),
                             verbose = FALSE)

lixels$continuous_density_abw <- nkde_abw$k * nrow(bike_accidents) * 1000

# extracting the local bandwidth
local_bw <- nkde_abw$events
buff_bw <- gBuffer(local_bw,width = local_bw$bw, byid=TRUE)
sel_buff <- buff_bw[c(1, 52, 20, 86, 14, 75, 126, 200, 177), ]

# mapping the elements
tm_shape(lixels,unit = 'm') +
  tm_lines(1.7, title.col = 'continuous NKDE density',
           col = 'continuous_density_abw', n = 7,
           style = 'jenks', palette = pal)+
tm_shape(sel_buff)+
  tm_borders("black", lwd = 1)+
  tm_layout(legend.outside = TRUE, inner.margins = 0,
            outer.margins = 0, legend.title.size = 1,
            legend.outside.position = "right",
            frame = FALSE)
```

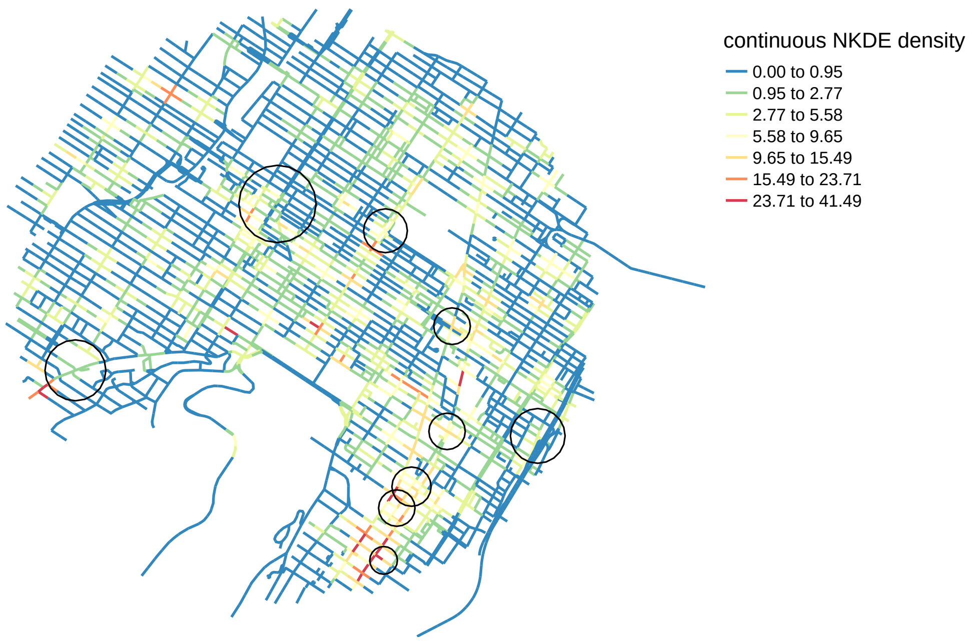


Figure 11: Estimated densities of the continuous NKDE for bike accidents in Montreal with an adaptive bandwidth.

Local Moran I

It has been proposed to combine the NKDE with the local Moran I autocorrelation index to identify significant hot spots or cold spots of events' intensity (Xie and Yan, 2013). Such an approach can be easily applied with **spNetwork**, which provides functions to calculate spatial weight matrices using network distances. We illustrate it here by creating a weight matrix w. w_{ij} is the spatial weight for observations i and j, calculated as the inverse of the network distance between them. Beyond 400 meters the weight is set to 0, and the matrix is row standardized. This example illustrates that **spNetwork** is well integrated with the actual ecosystem of R packages dedicated to spatial analysis.

```
library(spdep)

# calculating the spatial weight matrix
listw <- network_listw(origins = sample_pts,
                         lines = mtl_network,
                         maxdistance = 400, dist_func = "inverse",
                         matrice_type = "W",
                         grid_shape = c(1,1),digits = 2,
                         verbose = F, tol = 0.01)

# calculating the local Moran I values
locmoran<- localmoran(lixels$continuous_density,listw = listw)

# centering the NKDE values and calculating the spatially lagged variable
cent_density <- scale(lixels$continuous_density, scale = F)
lag_cent_density <- lag.listw(listw, cent_density)

# mapping the results
lixels$moran_quad <- case_when(
  cent_density > 0 & lag_cent_density > 0 & locmoran[,5]<=0.05 ~ "high-high",
  cent_density < 0 & lag_cent_density < 0 & locmoran[,5]<=0.05 ~ "low-low",
  cent_density > 0 & lag_cent_density < 0 & locmoran[,5]<=0.05 ~ "high-low",
  cent_density < 0 & lag_cent_density > 0 & locmoran[,5]<=0.05 ~ "low-high",
  locmoran[,5] > 0.05 ~ "not sign.",
)

colors <- c("high-high" = "#E63946",
            "high-low" = "#EC9A9A",
            "low-high" = "#A8DADC",
            "low-low" = "#457B9D",
            "not sign." = "black")

not_sign <- subset(lixels, lixels$moran_quad == "not sign.")
sign <- subset(lixels, lixels$moran_quad != "not sign.")

tm_shape(not_sign,unit = 'm') +
  tm_lines(1.2, title.col = 'Local Moran for the continuous NKDE',
           col = 'black') +
  tm_shape(sign,unit = 'm') +
  tm_lines(2, title.col = 'Local Moran for the continuous NKDE',
           col = 'moran_quad', palette = colors) +
  tm_layout(legend.outside = TRUE, inner.margins = 0,
            outer.margins = 0, legend.title.size = 1,
            legend.outside.position = "right",
            frame = FALSE)
```

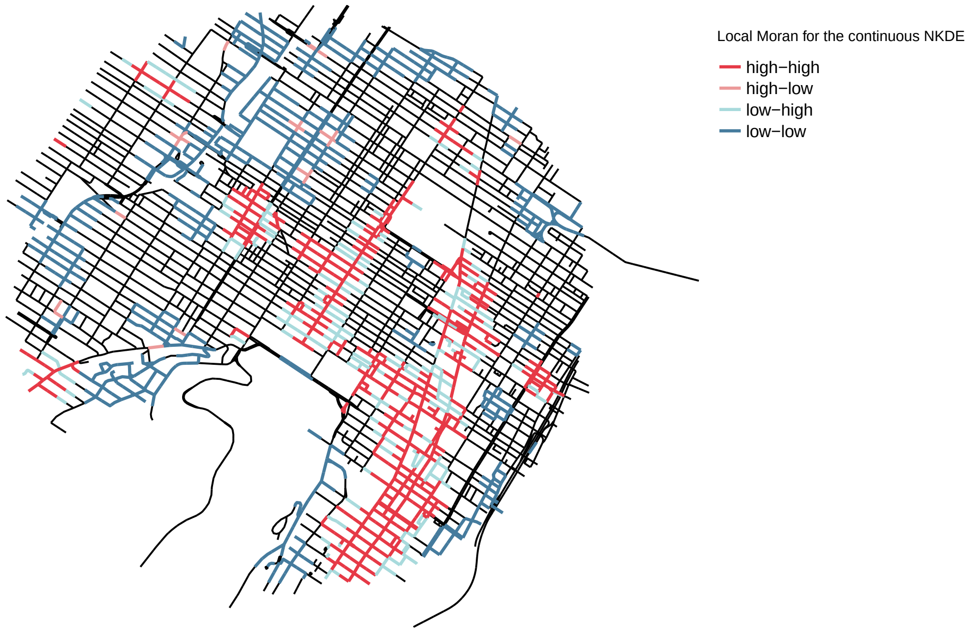


Figure 12: Local Moran I calculated on continuous NKDE.

Calculation time

In this final section, we investigate the calculation time of the three NKDE. For the first test, we used a complete version of the dataset presented above (but not included in the package). The network used covers the full Island of Montreal for a total of 47488 features (5877 km). The 1722 events are road accidents involving a cyclist between 2017 and 2018. We ran the calculation of the three NKDE by using 1 to 8 cores and by splitting the study area with a grid ranging from 10 by 10 to 18 by 18. The calculations were executed on a computer equipped with an Intel (R) Xeon (R) Bronze 3104 CPU (12 cores 1.70 GHz) and 32 GO of RAM, with the Windows 10 operating system and the R version 4.0.2. Table 2 shows the calculation times.

	Duration (min)								
	Simple			Discontinuous			Continuous		
Cores	10 x 10	14 x 14	18 x 18	10 x 10	14 x 14	18 x 18	10 x 10	14 x 14	18 x 18
1	23:33	19:22	18:02	21:49	17:31	16:37	87:29	43:37	33:55
2	14:01	11:48	11:36	14:37	11:23	10:31	59:13	32:57	19:41
3	10:28	08:42	08:20	10:57	08:22	08:03	53:35	27:54	17:15
4	09:44	09:42	07:43	09:12	09:00	07:24	78:00	32:23	17:26
5	08:02	07:45	07:13	08:04	07:09	06:21	91:16	30:08	14:56
6	07:50	06:49	07:06	07:01	06:51	06:33	93:05	35:58	12:19
7	07:38	06:43	06:39	06:38	06:36	06:24	53:06	38:17	14:20
8	06:54	06:37	07:06	07:06	06:38	06:56	55:52	33:32	18:47

Table 2: Durations of the first setting.

As one can see, the durations are very similar for the simple and the discontinuous methods. Because of the backward adjustment of the density, the continuous method is much more time-consuming. Using multiprocessing is an efficient way to reduce calculation time. In our setting, the biggest gain is achieved with three cores. Beyond three, the gains are more negligible. Dividing the study area with a finer grid contributes to reducing calculation time to a smaller extent.

For the second test, we selected a single event in the middle of the complete study area. We then constructed several study areas by selecting all events and all lines within subsequent radiuses from 500 m to 7000 m. For each iteration, only one core was used and a grid with a 5 x 5 shape. The results are presented in Table 3. We can observe that the calculation time increases slowly for the simple and the discontinuous NKDE, but the increase for the continuous NKDE is sharper. These results show that the proposed package can calculate the three NKDE with reasonable calculation time, even for datasets covering a full city.

			Duration (min)		
Distance	Number of events	Number of lines	Simple	Discontinuous	Continuous
1000	97	447	00:23	00:16	00:16
1500	198	948	00:41	00:27	00:28
2000	303	1648	00:58	00:40	00:43
2500	486	2581	01:25	00:59	01:03
3000	652	4010	02:01	01:30	04:47
3500	797	5630	02:39	02:03	13:20
4000	905	7032	03:13	02:21	14:06
4500	989	8497	03:44	02:53	16:17
500	22	138	00:13	00:10	00:10
5000	1097	10432	04:30	03:40	18:23
5500	1170	11913	05:23	04:27	22:18
6000	1230	13196	06:04	05:06	25:23
6500	1276	14674	06:44	05:46	32:49
7000	1332	16373	07:31	06:30	34:55

Table 3: Durations of the second setting.

Concluding remarks

The Network Kernel Density Estimate (NKDE) is an extension of the Kernel Density Estimate (KDE). Numerous spatial phenomena are constrained on a network like road accidents, crimes reported on streets, leaks in pipes, breaks in a wiring network, bird nests along a river, etc. Analyzing these datasets with a classical KDE is unsatisfactory because densities are evaluated at places where the events cannot occur (outside the network). The Euclidean distance underestimates the distance between objects on the network, and a network is not a homogeneous space. Tang et al. (2013) argue that in practical applications, the differences between the classical KDE and NKDE are small, the latter providing only more accurate and detailed results. However, our brief comparison suggests that the simple NKDE (the closest to the classical KDE) systematically overestimates densities compared to the continuous and discontinuous NKDE. This is in line with results obtained when comparing the planar and network K-functions (Yamada and Thill, 2004). The NKDE method remains quite inaccessible because of the lack of open-source implementation. The **spNetwork** package introduced here proposes to fill this gap by implementing it with R, taking advantage of its rich geospatial ecosystem (*Spatial*).
Three different NKDE are currently available in the package: the simple, the discontinuous, and the continuous NKDE. The first one is an early attempt to generalize the classical KDE to network spaces. It produces biased results and is not a true kernel, but it is intuitive from a purely geographical point of view. The discontinuous and continuous NKDE are true kernels and provide unbiased density estimates. We discussed in this paper their respective limitations and advantages. Two complementary features are proposed by **spNetwork**: Diggle's edge correction and adaptive bandwidths based on Abramson's rule. The **spNetwork** package uses several techniques to reduce computation time and memory use for the NKDE: dividing the study area into smaller chunks, multiprocessing, compiled C++ functions, and spatial indexes. However, for large datasets with large bandwidths the overall process remains long. Consequently, no methods are currently proposed for automatic bandwidth selection. In the future, we plan to add in the package other methods for PPA on networks like the K-function, nearest neighbor analysis and reticular distance weight matrices.

Bibliography

I. S. Abramson. On bandwidth variation in kernel estimates-a square root law. *The annals of Statistics*, pages 1217–1223, 1982. URL `https://www.jstor.org/stable/2240724`. [p563, 571]

A. Baddeley and R. Turner. spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. URL `http://www.jstatsoft.org/v12/i06/`. [p561]

A. Baddeley, E. Rubak, and R. Turner. *Spatial point patterns: methodology and applications with R*. CRC press, 2015. [p561]

A. J. Baddeley, R. Turner, et al. Spatstat: An r package for analyzing spatial point pattens, 2004. [p561]

G. Becker. *SearchTrees: Spatial Search Trees*, 2012. URL `https://CRAN.R-project.org/package=SearchTrees`. R package version 0.5.2. [p568]

H. Bengtsson. *future: Unified Parallel and Distributed Processing in R for Everyone*, 2020a. URL `https://CRAN.R-project.org/package=future`. R package version 1.17.0. [p567]

H. Bengtsson. *future.apply: Apply Function to Elements in Parallel using Futures*, 2020b. URL `https://CRAN.R-project.org/package=future.apply`. R package version 1.6.0. [p568]

R. Bivand and N. Lewin-Koh. *maptools: Tools for Handling Spatial Objects*, 2020. URL `https://CRAN.R-project.org/package=maptools`. R package version 1.0-1. [p561]

R. Bivand and C. Rundel. *rgeos: Interface to Geometry Engine - Open Source ('GEOS')*, 2020. URL `https://CRAN.R-project.org/package=rgeos`. R package version 0.5-3. [p561]

R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*, 2020. URL `https://CRAN.R-project.org/package=rgdal`. R package version 1.5-12. [p561]

G. Borruso. Network density and the delimitation of urban areas. *Transactions in GIS*, 7(2):177–191, 2003. URL `https://doi.org/10.1111/1467-9671.00139`. [p561]

G. Csardi and T. Nepusz. *igraph: Network Analysis and Visualization*, 2020. URL `https://CRAN.R-project.org/package=igraph`. R package version 1.2.5. [p561]

P. Diggle. A kernel method for smoothing point process data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 34(2):138–147, 1985. URL `https://doi.org/10.2307/2347366`. [p563]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2020a. URL `https://CRAN.R-project.org/package=Rcpp`. R package version 1.0.5. [p561]

D. Eddelbuettel, R. Francois, D. Bates, and B. Ni. *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*, 2020b. URL `https://CRAN.R-project.org/package=RcppArmadillo`. R package version 0.9.900.1.0. [p568]

ESRI. *ArcGIS Desktop: Release 10.5. 1*, 2017. [p561]

B. Flahaut, M. Mouchart, E. San Martin, and I. Thomas. The local spatial autocorrelation and the kernel method for identifying black zones: A comparative approach. *Accident Analysis & Prevention*, 35(6):991–1004, 2003. URL `https://doi.org/10.1016/S0001-4575(02)00107-0`. [p561]

M.-H. Hwang and A. Winslow. User manual for geodanet: spatial analysis on undirected networks, 2012. [p561]

U. Ligges and J. Fox. How can i avoid this loop or make it faster?, 2008. URL `http://www.r-project.org/doc/Rnews/Rnews_2008-1.pdf`. [p568]

G. McSwiggan, A. Baddeley, and G. Nair. Kernel density estimation on a linear network. *Scandinavian Journal of Statistics*, 44(2):324–345, 2017. URL `https://doi.org/10.1111/sjos.12255`. [p561]

A. Okabe and T. Satoh. Uniform network transformation for points pattern analysis on a non-uniform network. *Journal of Geographical Systems*, 8(1):25–37, 2006. URL `https://doi.org/10.1007/s10109-005-0009-2`. [p561]

A. Okabe and K. Sugihara. *Spatial analysis along networks: statistical and computational methods*. John Wiley & Sons, 2012. [p564, 566]

A. Okabe, K.-I. Okunuki, and S. Shiode. The sanet toolbox: new methods for network spatial analysis. *Transactions in GIS*, 10(4):535–550, 2006. URL `https://doi.org/10.1111/j.1467-9671.2006.01011.x`. [p561]

A. Okabe, T. Satoh, and K. Sugihara. A kernel density estimation method for networks, its computational method and a gis-based tool. *International Journal of Geographical Information Science*, 23(1):7–32, 2009. URL `https://doi.org/10.1080/13658810802475491`. [p561, 565, 566]

D. O'Sullivan and D. W. Wong. A surface-based approach to measuring spatial segregation. *Geographical analysis*, 39(2):147–168, 2007. URL https://doi.org/10.1111/j.1538-4632.2007.00699.x. [p563]

E. Pebesma and R. Bivand. *sp: Classes and Methods for Spatial Data*, 2020. URL https://CRAN.R-project.org/package=sp. R package version 1.4-2. [p561]

S. Porta, E. Strano, V. Iacoviello, R. Messora, V. Latora, A. Cardillo, F. Wang, and S. Scellato. Street centrality and densities of retail and services in bologna, italy. *Environment and Planning B: Planning and design*, 36(3):450–465, 2009. URL https://doi.org/10.1068/b34098. [p561]

QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2020. URL http://qgis.osgeo.org. [p567]

B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986. [p562]

T. Steenberghen, K. Aerts, and I. Thomas. Spatial clustering of events on a network. *Journal of Transport Geography*, 18(3):411–418, 2010. URL https://doi.org/10.1016/j.jtrangeo.2009.08.005. [p561]

Y. Tang, M. A. Knodler, and M.-H. Park. A comparative study of the application of the standard kernel density estimation and network kernel density estimation in crash hotspot identification. In *16th International Conference Road Safety on Four Continents. Beijing, China (RS4C 2013). 15-17 May 2013*. Statens väg-och transportforskningsinstitut, 2013. [p575]

B. A. Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*. Citeseer, 1993. [p563]

Z. Xie and J. Yan. Kernel density estimation of traffic accidents in a network space. *Computers, environment and urban systems*, 32(5):396–406, 2008. URL https://doi.org/10.1016/j.compenvurbsys.2008.05.001. [p561, 563, 564]

Z. Xie and J. Yan. Detecting traffic accident clusters with network kernel density estimation and local spatial statistics: an integrated approach. *Journal of transport geography*, 31:64–71, 2013. URL https://doi.org/10.1016/j.jtrangeo.2013.05.009. [p564, 572]

I. Yamada and J.-C. Thill. Comparison of planar and network k-functions in traffic accident analysis. *Journal of Transport Geography*, 12(2):149–158, 2004. URL https://doi.org/10.1016/j.jtrangeo.2003.10.006. [p575]

Jeremy Gelb
Institut National de la Recherche Scientifique, Urbanisation Culture et Société
385 Rue Sherbrooke E
Montréal, QC H2X 1E3
Canada (QC)
jeremy.gelb@inrs.ca

bssm: Bayesian Inference of Non-linear and Non-Gaussian State Space Models in R

by Jouni Helske and Matti Vihola

Abstract We present an R package **bssm** for Bayesian non-linear/non-Gaussian state space modeling. Unlike the existing packages, **bssm** allows for easy-to-use approximate inference based on Gaussian approximations such as the Laplace approximation and the extended Kalman filter. The package also accommodates discretely observed latent diffusion processes. The inference is based on fully automatic, adaptive Markov chain Monte Carlo (MCMC) on the hyperparameters, with optional importance sampling post-correction to eliminate any approximation bias. The package also implements a direct pseudo-marginal MCMC and a delayed acceptance pseudo-marginal MCMC using intermediate approximations. The package offers an easy-to-use interface to define models with linear-Gaussian state dynamics with non-Gaussian observation models and has an **Rcpp** interface for specifying custom non-linear and diffusion models.

Introduction

State space models (SSM) are a flexible class of latent variable models commonly used in analyzing time series data (cf. Durbin and Koopman, 2012). There are several packages available for state space modeling for R, especially for two special cases: a linear-Gaussian SSM (LGSSM) where both the observation and state densities are Gaussian with linear relationships with the states, and an SSM with discrete state space, which is sometimes called a hidden Markov model (HMM). These classes admit analytically tractable marginal likelihood functions and conditional state distributions (conditioned on the observations), making inference relatively straightforward. See, for example, Petris and Petrone (2011); Tusell (2011); Helske (2017); Helske and Helske (2019) for a review of some of the R packages dealing with these type of models. The present R package **bssm** is designed for Bayesian inference of general state space models with non-Gaussian and/or non-linear observational and state equations. The package's primary aim is to provide easy-to-use and fast functions for fully Bayesian inference with common time series models such as the basic structural time series model (Harvey, 1989) with exogenous covariates and simple stochastic volatility models. The package also accommodates custom non-linear models and discretized diffusion models.

When extending the state space modeling to non-linear or non-Gaussian models, some difficulties arise. As the likelihood is no longer analytically tractable, computing the latent state distributions and hyperparameter estimation of the model becomes more challenging. One general option is to use Markov chain Monte Carlo (MCMC) methods targeting the full joint posterior of hyperparameters and the latent states, for exampl,e by Gibbs sampling or Hamiltonian Monte Carlo. Unfortunately, the joint posterior is typically very high dimensional, and due to the strong autocorrelation structures of the state densities, the efficiency of such methods can be relatively poor. Another asymptotically exact approach is based on the pseudo-marginal particle MCMC approach (Andrieu et al., 2010), where the likelihood function and the state distributions are estimated using sequential Monte Carlo (SMC), i.e., the particle filter (PF) algorithm. Instead of computationally demanding Monte Carlo methods, approximation-based methods such as extended and unscented Kalman filters may be used, as well as Laplace approximations, which are provided for example by the **INLA** (Lindgren and Rue, 2015) R package. These approximations are computationally appealing but may lead to hard-to-quantify biases of the posterior.

Some of the R packages suitable for Bayesian state space modeling include **pomp** (King et al., 2016), **rbi** (Jacob and Funk, 2020), **nimbleSMC** (Michaud et al., 2020; NIMBLE Development Team, 2020), and **rstan** (Stan Development Team, 2020). With the package **pomp**, the user defines the model using R or C snippets for simulation from and evaluation of the latent state and observation level densities, allowing flexible model construction. The **rbi** package is an interface to LibBi (Murray, 2015), a standalone software with a focus on Bayesian state space modeling on high-performance computers. The **pomp** package provides several simulation-based inference methods mainly based on iterated filtering and maximum likelihood, whereas **rbi** is typically used for Bayesian inference via particle MCMC. For a more detailed comparison of differences of **rbi**/LibBi and **pomp** with examples; see (Funk and King, 2020). The **nimbleSMC** package contains some particle filtering algorithms which can be used in the general Nimble modeling system (de Valpine et al., 2017), whereas the **rstan** package provides an R interface to the Stan C++ package, a general statistical modeling platform (Carpenter et al., 2017).

The key difference to the aforementioned packages and the motivation behind the present **bssm** package is to combine the use of fast approximation-based methods with the Monte Carlo correction step, leading to computationally efficient and unbiased (approximation error free) inference of the joint posterior of hyperparameters and latent states, as suggested in (Vihola et al., 2020). In a nutshell, the method uses MCMC, which targets an approximate marginal posterior of the hyperparameters and an importance sampling type weighting which provides asymptotically exact inference on the joint posterior of hyperparameters and the latent states. In addition to this two-stage procedure, the **bssm** also supports delayed acceptance pseudo-marginal MCMC (Christen and Fox, 2005) using the approximations and direct pseudo-marginal MCMC. To our knowledge, importance sampling and delayed acceptance in this form are not available in other Bayesian state space modeling packages in R.

Supported models

We denote the sequence of observations $(y_1, \ldots, y_T)$ as y, and the sequence of latent state variables $(\alpha_1, \ldots, \alpha_T)$ as α. The latent states $\alpha_t \in \mathbb{R}^d$ are typically vector-valued, whereas we focus mainly on scalar observations $y_t \in \mathbb{R}$ (vector-valued observations are also supported, assuming conditional independence (given α_t) in case of non-Gaussian observations).

A general state space model consists of two parts: observation level densities $g_t^{(\theta)}(y_t|\alpha_t)$ and latent state transition densities $\mu_t^{(\theta)}(\alpha_{t+1}|\alpha_t)$. Typically, both $g_t^{(\theta)}$ and $\mu_t^{(\theta)}$ depend on unknown parameter vector θ for which we can define arbitrary prior $p(\theta)$.

In a linear-Gaussian SSM, both $g_t^{(\theta)}$ and $\mu_t^{(\theta)}$ are Gaussian densities, and they depend linearly on the current and previous state vectors, respectively. Section Models with linear-Gaussian state dynamics describes a common extension to these models supported by **bssm**, which relaxes the assumptions on observational density $g_t^{(\theta)}$, by allowing exponential family links and stochastic volatility models. While the main focus of **bssm** is in state space models with linear-Gaussian state dynamics, there is also support for more general non-linear models, discussed briefly in Section Other state space models. Section Using the **bssm** package describes how arbitrary models based on these definitions are constructed in **bssm**.

Models with linear-Gaussian state dynamics

The primary class of models supported by **bssm** consists of SSMs with linear-Gaussian state dynamics of form

$$\alpha_{t+1} = c_t + T_t\alpha_t + R_t\eta_t,$$

where $c_t \in \mathbb{R}^d$, $T_t \in \mathbb{R}^{d\times d}$, and $R_t \in \mathbb{R}^{d\times k}$ can depend on the unknown parameters θ and covariates. The noise terms $\eta_t \sim N(0, I_k)$ and $\alpha_1 \sim N(a_1, P_1)$ are independent. These state dynamics can be combined with the observational level density g_t of form

$$g_t(y_t|d_t + Z_t\alpha_t, \phi, u_t),$$

where parameters ϕ and the known vector u_t are distribution specific and can be omitted in some cases. Currently, following observational level distributions are supported:

- Gaussian distribution: $y_t = d_t + Z_t\alpha_t + H_t\epsilon_t$ with $\epsilon_t \sim N(0, I)$.
- Poisson distribution: $g_t(y_t|d_t + Z_t\alpha_t, u_t) = \text{Poisson}(u_t \exp(d_t + Z_t\alpha_t))$, where u_t is the known exposure at time t.
- Binomial distribution: $g_t(y_t|d_t + Z_t\alpha_t, u_t) = \text{B}(u_t, \text{logit}^{-1}(d_t + Z_t\alpha_t))$, where u_t is the number of trials, and $\text{logit}^{-1}(d_t + Z_t\alpha_t)$ is the probability of the success.
- Negative binomial distribution: $g_t(y_t|d_t + Z_t\alpha_t, \phi, u_t) = \text{NB}(\exp(d_t + Z_t\alpha_t), \phi, u_t)$, where $u_t \exp(d_t + Z_t\alpha_t)$ is the expected value, ϕ is the dispersion parameter, and u_t is a known offset term.
- Gamma distribution: $g_t(y_t|d_t + Z_t\alpha_t, \phi, u_t) = \text{Gamma}(\exp(d_t + Z_t\alpha_t), \phi, u_t)$, where $u_t \exp(d_t + Z_t\alpha_t)$ is the expected value, ϕ is the shape parameter, and u_t is a known offset term.
- Stochastic volatility model: $g_t(y_t|Z_t\alpha_t) = \exp(\alpha_t/2)\epsilon_t$, with $\epsilon_t \sim N(0,1)$. Here, the state dynamics is also fixed as $\alpha_{t+1} = \mu + \rho(\alpha_t - \mu) + \sigma_\eta\eta_t$, with $\eta_t \sim N(0,1)$ and $\alpha_1 \sim N(\mu, \sigma^2_\eta/(1-\rho^2))$.

For multivariate models, these distributions can be combined arbitrarily, except the stochastic volatility model case, which is currently handled separately. Also, for a fully Gaussian model, the observational level errors ϵ_t can be correlated across time series.

Other state space models

The general non-linear Gaussian model in the **bssm** has the following form:

$$\begin{aligned} y_t &= Z(t, \alpha_t, \theta) + H(t, \alpha_t, \theta)\epsilon_t, \\ \alpha_{t+1} &= T(t, \alpha_t, \theta) + R(t, \alpha_t, \theta)\eta_t, \\ \alpha_1 &\sim N(a_1(\theta), P_1(\theta)), \end{aligned}$$

with $t = 1, \ldots, n$, $\epsilon_t \sim N(0, \mathrm{I}_p)$, and $\eta \sim N(0, \mathrm{I}_k)$.

The **bssm** package also supports models where the state equation is defined as a continuous-time diffusion model of the form

$$\mathrm{d}\alpha_t = \mu(\alpha_t, \theta)\mathrm{d}t + \sigma(\alpha_t, \theta)\mathrm{d}B_t, \quad t \geq 0,$$

where B_t is a Brownian motion and where μ and σ are scalar-valued functions, with the univariate observation density $p(y_k|\alpha_k)$ defined at integer times $k = 1 \ldots, n$.

Inference methods

The main goal of **bssm** is to facilitate easy-to-use full Bayesian inference of the joint posterior $p(\alpha, \theta|y)$ for models discussed in Section Supported models. The inference methods implemented in **bssm** are based on a factorized approach where the joint posterior of hyperparameters θ and latent states α is given as

$$p(\alpha, \theta|y) \propto p(\theta)p(\alpha, y|\theta) = p(\theta)p(y|\theta)p(\alpha|y, \theta),$$

where $p(y|\theta)$ is the parameter marginal likelihood and $p(\alpha|y, \theta)$ is the smoothing distribution.

All the inference algorithms are based on a Markov chain Monte Carlo on the parameters θ, whose single iteration may be summarised as follows:

Algorithm 1 One iteration of MCMC algorithm for sampling $p(\theta|y)$.

```
1: Draw a proposal $\theta' \sim N(\theta^{i-1}, \Sigma_{i-1})$.
2: Calculate the (approximate) marginal likelihood $\hat{p}(y|\theta')$.
3: Accept the proposal with probability $\alpha := \min\left\{1, \frac{p(\theta')\hat{p}(y|\theta')}{p(\theta^{i-1})\hat{p}(y|\theta^{i-1})}\right\}$.
4: If the proposal $\theta'$ is accepted, set $\theta^i = \theta'$. Otherwise, set $\theta^i = \theta^{i-1}$.
5: Adapt the proposal covariance matrix $\Sigma_{i-1} \to \Sigma_i$.
```

The adaptation step 5 in **bssm** currently implements the robust adaptive Metropolis algorithm (Vihola, 2012) with a fixed target acceptance rate (0.234 by default) provided by the **ramcmc** package (Helske, 2016). The (approximate) marginal likelihood $\hat{p}(y|\theta)$ takes different forms, leading to different inference algorithms discussed below.

Direct inference: marginal algorithm and particle MCMC

The simplest case is with a linear-Gaussian SSM, where we can use the exact marginal likelihood $\hat{p}(y|\theta) = p(y|\theta)$, in which case Algorithm 1 reduces to (an adaptive) random-walk Metropolis algorithm targeting the posterior marginal of the parameters θ. Inference from the full posterior may be made using the simulation smoothing algorithm (Durbin and Koopman, 2002) conditional to the sampled hyperparameters.

The other 'direct' option, which can be used with any model, is using the bootstrap particle filter (BSF) (Gordon et al., 1993), which leads to a *random* $\hat{p}(y|\theta)$ which is an unbiased estimator of $p(y|\theta)$. In this case, Algorithm 1 reduces to (an adaptive) particle marginal Metropolis-Hastings (Andrieu et al., 2010). The full posterior inference is achieved simultaneously, by picking particle trajectories based on their ancestries as in the filter-smoother algorithm (Kitagawa, 1996). Note that with BSF, the desired acceptance rate needs to be lower, depending on the number of particles used (Doucet et al., 2015).

Approximate inference: Laplace approximation and the extended Kalman filter

The direct BSF discussed above may be used with any non-linear and/or non-Gaussian model but may be slow and/or poor mixing. To alleviate this, **bssm** provides computationally efficient (intermediate) approximate inference in case of non-Gaussian observation models in Section Models with linear-Gaussian state dynamics and in case of non-linear dynamics in Section Other state space models.

With non-Gaussian models with linear-Gaussian dynamics, we use an approximating Gaussian model $\tilde{p}(y, \alpha|\theta)$ which is a Laplace approximation of $p(\alpha, y|\theta)$ following (Durbin and Koopman, 2000). We write the likelihood as follows

$$p(y|\theta) = \int p(\alpha, y|\theta) \mathrm{d}\alpha = \tilde{p}(y|\theta) E\left[\frac{p(y|\alpha,\theta)}{\tilde{p}(y|\alpha,\theta)}\right],$$

where $\tilde{p}(y|\theta)$ is the likelihood of the Laplace approximation, and the expectation is taken with respect to its conditional $\tilde{p}(\alpha|y,\theta)$ (Durbin and Koopman, 2012). Indeed, denoting $\hat{\alpha}$ as the mode of $\tilde{p}(\alpha|\theta, y)$, we may write

$$\log p(y|\theta) = \log \tilde{p}(y|\theta) + \log \frac{p(y|\hat{\alpha},\theta)}{\tilde{p}(y|\hat{\alpha},\theta)} + \log E\left[\frac{p(y|\alpha,\theta)/p(y|\hat{\alpha},\theta)}{\tilde{p}(y|\alpha,\theta)/\tilde{p}(y|\hat{\alpha},\theta)}\right].$$

If $\tilde{p}$ resembles p with typical values of α, the latter logarithm of expectation is zero. We take $\hat{p}(y|\theta)$ as the expression on the right, dropping the expectation.

When $\hat{p}$ is approximate, the MCMC algorithm targets an approximate posterior marginal. Approximate full inference may be done analogously as in the case of the previous section by simulating trajectories conditional to the sampled parameter configurations θ^i. We believe that approximate inference is often good enough for model development, but we strongly recommend using post-correction as discussed in Section Post-processing by importance weighting to check the validity of the final inference.

In addition to these algorithms, **bssm** also supports $\hat{p}(y|\theta)$ based on the extended KF (EKF) or iterated EKF (IEKF) (Jazwinski, 1970), which can be used for models with non-linear dynamics. Approximate smoothing based on (iterated) EKF is also supported. It is also possible to perform direct inference which instead of the BSF, employs particle filter based on EKF (Van Der Merwe et al., 2001).

Post-processing by importance weighting

The approximate inference methods of the previous section are computationally efficient but come with a bias. The **bssm** implements importance-sampling type post-correction as discussed in (Vihola et al., 2020). Indeed, having MCMC samples (θ^i) from the approximate posterior, we may produce (random) weights and latent states, such that the weighted samples form estimators which are consistent with respect to the true posterior $p(\alpha, \theta|y)$.

The primary approach which we recommend for post-correction is based on a "ψ-APF", — a particle filter using intermediate Gaussian approximations in the previous section. In essence, this particle filter employs the dynamics and a look-ahead strategy coming from the approximation, which leads to low-variance estimators; see (Vihola et al., 2020) and package vignettes[1] for a more detailed description. Naturally, ψ-APF can also be used in place of BSF in the direct inference of Section Direct inference: marginal algorithm and particle MCMC.

Direct inference using approximation-based delayed acceptance

An alternative to approximate MCMC and post-correction, **bssm** also supports an analogous delayed acceptance method (Christen and Fox, 2005; Banterle et al., 2019) (here denoted by DA-MCMC). This algorithm is similar to 1, but in the case of "acceptance", it leads to second-stage acceptance using the same weights as the post-correction would; see (Vihola et al., 2020) for details. Note that as in the direct approach for non-Gaussian/non-linear models, the desired acceptance rate with DA-MCMC should be lower than the default 0.234.

The DA-MCMC also leads to consistent posterior estimators and often outperforms the direct particle marginal Metropolis-Hastings. However, empirical findings (Vihola et al., 2020) and theoretical considerations (Franks and Vihola, 2020) suggest that approximate inference with post-correction may often be preferable. The **bssm** supports parallelization with post-correction using OpenMP, which may further promote the latter.

[1]`https://cran.r-project.org/package=bssm/vignettes/psi_pf.html`

Inference with diffusion state dynamics

For general continuous-time diffusion models, the transition densities are intractable. The **bssm** uses Millstein time-discretization scheme for approximate simulation, and inference is based on the corresponding BSF. Fine time-discretization mesh gives less bias than the coarser one, with increased computational complexity. The DA and IS approaches can be used to speed up the inference by using coarse discretization in the first stage and then using more fine mesh in the second stage. For comparison of DA and IS approaches in the case of geometric Brownian motion model, see (Vihola et al., 2020).

Using the bssm package

The main functions of **bssm** related to the MCMC sampling, approximations, and particle filtering are written in C++, with help the of **Rcpp** (Eddelbuettel and François, 2011) and **RcppArmadillo** (Eddelbuettel and Sanderson, 2014) packages. On the R side, the package uses S3 methods to provide a relatively unified workflow independent of the type of model one is working with. The model building functions such as `bsm_ng` and `svm` are used to construct the model objects of the same name, which can then be passed to other methods, such as `logLik` and `run_mcmc`, which compute the log-likelihood value and run MCMC algorithm, respectively. We will now briefly describe the main functionality of **bssm**. For more detailed descriptions of different functions and their arguments, see the corresponding documentation in R and the package vignettes.

Constructing the model

For models with linear-Gaussian state dynamics, **bssm** includes some predefined models such as `bsm_lg` and `bsm_ng` for univariate Gaussian and non-Gaussian structural time series models with external covariates, for which the user only needs to supply the data and priors for unknown model parameters. In addition, **bssm** supports general model building functions `ssm_ulg`, `ssm_mlg` for custom univariate and multivariate Gaussian models, and `ssm_ung` and `ssm_mng` for their non-Gaussian counterparts. For these models, users need to supply their own R functions for the evaluation of the log prior density and for updating the model matrices given the current value of the parameter vector θ. It is also possible to avoid defining the matrices manually by leveraging the formula interface of the **KFAS** package (Helske, 2017) together with `as_bssm` function which converts the KFAS model to a **bssm** equivalent model object. This is especially useful in the case of complex multivariate models with covariates.

As an example, consider a Gaussian local linear trend model of the form

$$\begin{aligned} y_t &= \mu_t + \epsilon_t, \\ \mu_{t+1} &= \mu_t + \nu_t + \eta_t, \\ \nu_{t+1} &= \nu_t + \xi_t, \end{aligned}$$

with zero-mean Gaussian noise terms $\epsilon_t, \eta_t, \xi_t$ with unknown standard deviations. Using the time series of the mean annual temperature (in Fahrenheit) in New Haven, Connecticut, from 1912 to 1971 (available in the `datasets` package) as an example, this model can be built with `bsm` function as

```
library("bssm")
data("nhtemp", package = "datasets")
prior <- halfnormal(1, 10)
bsm_model <- bsm_lg(y = nhtemp, sd_y = prior, sd_level = prior, sd_slope = prior)
```

Here, we use a helper function, `halfnormal`, which defines half-Normal prior distribution for the standard deviation parameters, with the first argument defining the initial value of the parameter and the second defines the scale parameter of the half-Normal distribution. Other prior options are `normal`, `tnormal` (truncated normal), `gamma`, and `uniform`.

As an example of a multivariate model, consider bivariate Poisson model with latent random walk model, defined as

$$\begin{aligned} y_{i,t} &\sim \text{Poisson}(\exp(x_t)), \quad i = 1,2, \\ x_{t+1} &= x_t + \eta_t, \end{aligned}$$

with $\eta_t \sim N(0, \sigma^2)$, and prior $\sigma \sim \text{Gamma}(2, 0.01)$. This model can be built with `ssm_mng` function as

```
# Generate observations
set.seed(1)
```

```
x <- cumsum(rnorm(50, sd = 0.2))
y <- cbind(
  rpois(50, exp(x)),
  rpois(50, exp(x)))

# Log prior density function
prior_fn <- function(theta) {
  dgamma(theta, 2, 0.01, log = TRUE)
}

# Model parameters from hyperparameters
update_fn <- function(theta) {
  list(R = (theta, c(1, 1, 1)))
}

# define the model
mng_model <- ssm_mng(y = y, Z = matrix(1,2,1), T = 1,
  R = 0.1, P1 = 1, distribution = "poisson",
  init_theta = 0.1,
  prior_fn = prior_fn, update_fn = update_fn)
```

Here, the user-defined functions prior_fn and update_fn define the log-prior for the model and how the model components depend on the hyperparameters θ, respectively.

For models where the state equation is no longer linear-Gaussian, we use a pointer-based interface by defining all model components as well as functions defining the Jacobians of $Z(\cdot)$ and $T(\cdot)$ needed by the extended Kalman filter as C++ snippets. The general non-linear Gaussian model can be defined with the function ssm_nlg. Discretely observed diffusion models where the state process is assumed to be a continuous stochastic process can be constructed using the ssm_sde function, which takes pointers to C++ functions defining the drift, diffusion, the derivative of the diffusion function, and the log-densities of the observations and the prior. As an example of the latter, let us consider an Ornstein–Uhlenbeck process

$$d\alpha_t = \rho(\nu - \alpha_t)dt + \sigma dB_t,$$

with parameters $\theta = (\phi, \nu, \sigma) = (0.5, 2, 1)$ and the initial condition $\alpha_0 = 1$. For observation density, we use Poisson distribution with parameter $\exp(\alpha_k)$. We first simulate a trajectory $x_0, \ldots, x_n$ using the sde.sim function from the **sde** package (Iacus, 2016) and use that for the simulation of observations y:

```
library("sde")
x <- sde.sim(t0 = 0, T = 100, X0 = 1, N = 100,
  drift = expression(0.5 * (2 - x)),
  sigma = expression(1),
  sigma.x = expression(0))
y <- rpois(100, exp(x[-1]))
```

We then compile and build the model as

```
Rcpp::sourceCpp("ssm_sde_template.cpp")
pntrs <- create_xptrs()
sde_model <- ssm_sde(y, pntrs$drift, pntrs$diffusion,
  pntrs$ddiffusion, pntrs$obs_density, pntrs$prior,
  c(0.5, 2, 1), 1, FALSE)
```

The templates for the C++ functions for SDE and non-linear Gaussian models can be found from the package vignettes on the CRAN[2].

Markov chain Monte Carlo in bssm

The main purpose of the **bssm** is to allow computationally efficient MCMC-based inference for various state space models. For this task, a method run_mcmc can be used. The function takes several arguments, depending on the model class, but for many of these, default values are provided. For linear-Gaussian models, we only need to supply the number of iterations. Using the previously created local linear trend model for the New Haven temperature data of Section Constructing the model, we run an MCMC with 100,000 iterations where the first 10,000 is discarded as a burn-in (burn-in phase is also used for the adaptation of the proposal distribution):

[2]https://CRAN.R-project.org/package=bssm

```
mcmc_bsm <- run_mcmc(bsm_model, iter = 1e5, burnin = 1e4)
```

The `print` method for the output of the MCMC algorithms gives a summary of the results, and detailed summaries for θ and α can be obtained using `summary` function. For all MCMC algorithms, **bssm** uses so-called jump chain representation of the Markov chain $X_1, \ldots, X_n$, where we only store each accepted X_k and the number of steps we stayed on the same state. So for example if $X_{1:n} = (1,2,2,1,1,1)$, we present such chain as $\tilde{X} = (1,2,1)$, $N = (1,2,3)$. This approach reduces the storage space and makes it more computationally efficient to use importance sampling type correction algorithms. One drawback of this approach is that the results from the MCMC run correspond to weighted samples from the target posterior, so some of the commonly used postprocessing tools need to be adjusted. Of course, in the case of other methods than IS-weighting, the simplest option is to just expand the samples to a typical Markov chain using the stored counts N. This can be done using the function `expand_sample`, which returns an object of class `"mcmc"` of the **coda** package (Plummer et al., 2006) (thus the plotting and diagnostic methods of **coda** can also be used). We can also directly transform the posterior samples to a `"data.frame"` object by using `as.data.frame` method for the MCMC output (for IS-weighting, the returned data frame contains additional column `weights`). This is useful, for example, for visualization purposes with the **ggplot2** (Wickham, 2016) package:

```
library("ggplot2")
d <- as.data.frame(mcmc_bsm, variable = "theta")
ggplot(d, aes(x = value)) +
  geom_density(bw = 0.1, fill = "#9ebcda") +
  facet_wrap(~ variable, scales = "free") +
  theme_bw()
```

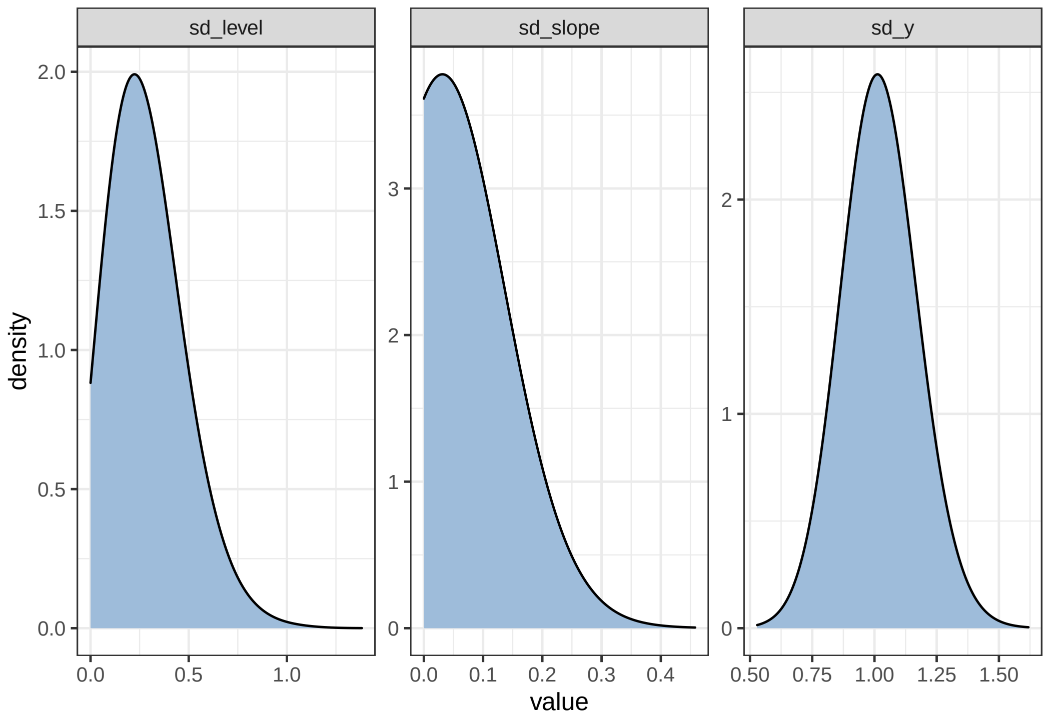


Figure 1: Posterior densities of hyperparameters θ of the linear-Gaussian model for nhtemp data.

Figure 1 shows the estimated posterior densities of the three standard deviation parameters of the model. The relatively large observational level standard deviation σ_y suggests that the underlying latent temperature series is much smoother than the observed series, which can also be seen from Figure 2, which shows the original observations (black dots) spread around the estimated temperature series (solid line). This figure was drawn using **dplyr** (Wickham et al., 2021) and **ggplot2** with the following code:

```
library("dplyr")
d <- as.data.frame(mcmc_bsm, variable = "states")
summary_y <- d %>%
  filter(variable == "level") %>%
  group_by(time) %>%
    summarise(mean = mean(value),
      lwr = quantile(value, 0.025),
      upr = quantile(value, 0.975))

ggplot(summary_y, aes(x = time, y = mean)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.25) +
  geom_line() +
  geom_point(data = data.frame(mean = nhtemp,
    time = time(nhtemp))) +
  theme_bw() + xlab("Year") +
  ylab("Mean annual temperature in New Haven")
```

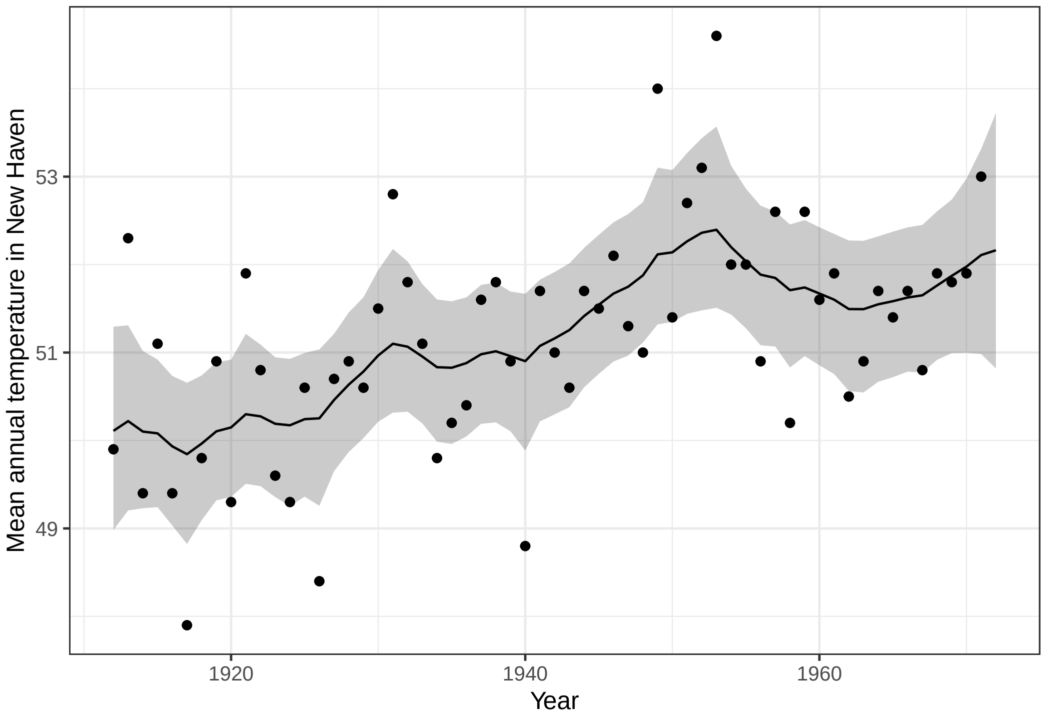


Figure 2: Observed annual average temperatures in New Haven (black dots) and predicted mean (solid line) with 95% prediction intervals (grey ribbon) from **bssm**.

For non-Gaussian models, the default MCMC algorithm is an approximate inference based on Laplace approximation combined with importance sampling post-correction. It is also possible to perform first approximate MCMC using the argument `mcmc_type = "approx"` and then perform the post-correction step using the results from the approximate MCMC. In doing so, we can also use the function `suggest_N` to find a suitable number of particles N for ψ-APF in the spirit of Doucet et al. (2015):

```
out_approx <- run_mcmc(mng_model, mcmc_type = "approx", iter = 50000)
est_N <- suggest_N(mng_model, out_approx)
out_exact <- post_correct(mng_model, out_approx, particles = est_N$N)
```

The function `suggest_N` computes the standard deviation of the logarithm of the post-correction weights (i.e., the random part of log-likelihood of ψ-APF) at the approximate MAP estimator of θ using a range of N and returns a list with component `N` which is the smallest number of particles where the standard deviation was less than one. For small and moderate problems, typically, 10-20 particles are enough.

Filtering and smoothing

The **bssm** package also offers separate methods for performing (approximate) state filtering and smoothing which may be useful in some custom settings.

For LGSSM, methods `kfilter` and `smoother` perform Kalman filtering and smoothing. For non-Gaussian models with linear-Gaussian dynamics, approximate filtering and smoothing estimates can be obtained by calls to kfilter and smoother. These functions first construct an approximating Gaussian model for which the Kalman filter/smoother is then applied. For non-linear models defined by nlg_ssm we can run approximate filtering using the extended Kalman filter with the function ekf, the unscented Kalman filter with the function ukf, or the iterated EKF by changing the argument iekf_iter of the ekf function. Function ekf_smoother can be used for smoothing based on EKF/IEKF.

For particle filtering, the **bssm** package supports a general bootstrap particle filter for all model classes of the **bssm** (function bootstrap_filter). For "nlg_ssm", extended Kalman particle filtering (Van Der Merwe et al., 2001) is also supported (function ekpf_filter). For particle smoothing, function particle_smoother with the smoothing based on BSF is available for all models. In addition, ψ-APF (using argument method = "psi") is available for all models except for "ssm_sde" class. Currently, only the filter-smoother approach (Kitagawa, 1996) for particle smoothing is supported.

Comparison of IS-MCMC and HMC

Vihola et al. (2020) compared the computational efficiency of delayed acceptance MCMC and importance sampling type MCMC approaches in various settings. Here we make a small experiment comparing the generic Hamiltonian Monte Carlo using the NUTS sampler (Hoffman and Gelman, 2014) with **rstan** and IS-MCMC with **bssm**. Given that the **bssm** package is specialized for state space models whereas Stan is a general purpose tool suitable for a wider range of problems, it is to be expected that **bssm** performs better in terms of computational efficiency. The purpose of this comparison is to illustrate this fact, i.e., that there is still demand for specialized algorithms for various types of statistical models. For the complete code of the experiment, see supplementary materials.

We consider the case of a random walk with drift model with negative binomial observations and some known covariate x_t, defined as

$$\begin{aligned}
y_t &\sim \mathrm{NB}(\exp(\beta x_t + \mu_t), \phi), \\
\mu_{t+1} &= \mu_t + \nu_t + \eta_t, \\
\nu_{t+1} &= \nu_t,
\end{aligned}$$

with zero-mean Gaussian noise term η_t with unknown standard deviation σ_μ. Based on this we simulate one realization of y and x with $n = 200$, $\phi = 5$, $\beta = -0.9$, $\nu = 0.01$, $\sigma_\mu = 0.1$.

For the IS approach, we use ng_bsm function for model building, with prior variances 100 and 0.01 for the initial states μ_1 and ν_1. For hyperparameters, we used a fairly uninformative half-Normal distribution with a standard deviation of 0.5 for σ_μ and 0.1 for σ_ν. We then ran the IS-MCMC algorithm with run_mcmc using a burn-in phase of length 10,000 and ran 50,000 iterations after the burn-in, with 10 particles per SMC.

Using the same setup, we ran the MCMC with **rstan** using 15,000 iterations (with the first 5000 used for warm-up). Note that in order to avoid sampling problems, it was necessary to tweak the default control parameters of the sampler (see Appendix).

Table 1 shows the results. We see both methods produce identical results (within the Monte Carlo error), but while **rstan** produces similar Monte Carlo standard errors with a smaller amount of total iterations than **bssm**, the total computation time of **rstan** is almost 80 times higher than with **bssm** (58 minutes versus 45 seconds), which suggests that for these types of problems it is highly beneficial to take advantage of the known model structure and available approximations versus general Bayesian software such as Stan which makes no distinction between latent states α and hyperparameters θ.

	bssm			**rstan**		
	Mean	SD	MCSE	Mean	SD	MCSE
σ_μ	0.092	0.037	9×10^{-4}	0.090	0.036	9×10^{-4}
σ_ν	0.003	0.003	5×10^{-5}	0.003	0.003	7×10^{-5}
ϕ	5.392	0.910	2×10^{-2}	5.386	0.898	1×10^{-2}
β	-0.912	0.056	1×10^{-3}	-0.911	0.056	7×10^{-4}
μ_{200}	6.962	0.346	5×10^{-3}	6.965	0.349	4×10^{-3}
ν_{200}	0.006	0.020	3×10^{-4}	0.006	0.019	2×10^{-4}

Table 1: Estimates of posterior mean, standard deviation and Monte Carlo standard error of the mean for hyperparameters θ and latent states for last time point for the example model.

Conclusions

State space models are a flexible tool for analyzing a variety of time series data. Here, we introduced the R package **bssm** for fully Bayesian state space modeling for a large class of models with several alternative MCMC sampling strategies. All computationally intensive parts of the package are implemented with C++ with parallel computation support for IS-MCMC making it an attractive option for many common models where relatively accurate Gaussian approximations are available.

Compared to early versions of the **bssm** package, the option to define R functions for model updating and prior evaluation has lowered the bar for analyzing custom models. The package is also written in a way that it is relatively easy to extend to new model types similar to current `bsm_lg` in the future. The **bssm** package could be expanded to allow other proposal adaptation schemes such as the adaptive Metropolis algorithm by Haario et al. (2001), as well as support for multivariate SDE models and automatic differentiation for EKF-type algorithms.

Acknowledgements

This work has been supported by the Academy of Finland research grants 284513, 312605, 315619, 311877, and 331817.

Bibliography

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of Royal Statistical Society B*, 72(3):269–342, 2010. [p578, 580]

M. Banterle, C. Grazian, A. Lee, and C. P. Robert. Accelerating Metropolis-Hastings algorithms by delayed acceptance. *Foundations of Data Science*, 1(2):103, 2019. URL `https://doi.org/10.3934/fods.2019005`. [p581]

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1): 1–32, 2017. URL `https://doi.org/10.18637/jss.v076.i01`. [p578]

J. A. Christen and C. Fox. Markov chain Monte Carlo using an approximation. *Journal of Computational and Graphical Statistics*, 14(4):795–810, 2005. URL `https://doi.org/10.1198/106186005X76983`. [p579, 581]

P. de Valpine, D. Turek, C. Paciorek, C. Anderson-Bergman, D. Temple Lang, and R. Bodik. Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics*, 26:403–413, 2017. URL `https://doi.org/10.1080/10618600.2016.1172487`. [p578]

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313, 03 2015. ISSN 0006-3444. URL `https://doi.org/10.1093/biomet/asu075`. [p580, 585]

J. Durbin and S. J. Koopman. Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives. *Journal of Royal Statistical Society B*, 62:3–56, 2000. [p581]

J. Durbin and S. J. Koopman. A simple and efficient simulation smoother for state space time series analysis. *Biometrika*, 89:603–615, 2002. [p580]

J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, New York, 2nd edition, 2012. [p578, 581]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL `https://doi.org/10.18637/jss.v040.i08`. [p582]

D. Eddelbuettel and C. Sanderson. Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. URL `http://doi.org/10.1016/j.csda.2013.02.005`. [p582]

J. Franks and M. Vihola. Importance sampling correction versus standard averages of reversible MCMCs in terms of the asymptotic variance. *Stochastic Processes and their Applications*, 130(10): 6157 – 6183, 2020. ISSN 0304-4149. URL http://www.sciencedirect.com/science/article/pii/S0304414919304053. [p581]

S. Funk and A. A. King. Choices and trade-offs in inference with infectious disease models. *Epidemics*, 30:100383, 2020. ISSN 1755-4365. URL https://doi.org/10.1016/j.epidem.2019.100383. [p578]

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113, 1993. [p580]

H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. URL https://doi.org/10.2307/3318737. [p587]

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1989. [p578]

J. Helske. *ramcmc: Robust Adaptive Metropolis Algorithm*, 2016. URL https://CRAN.R-project.org/package=ramcmc. R package version 0.1.0-1.1. [p580]

J. Helske. KFAS: Exponential family state space models in R. *Journal of Statistical Software*, 78(10):1–39, 2017. URL https://doi.org/10.18637/jss.v078.i10. [p578, 582]

S. Helske and J. Helske. Mixture hidden Markov models for sequence data: The seqHMM package in R. *Journal of Statistical Software*, 88(3):1–32, 2019. URL https://doi.org/10.18637/jss.v088.i03. [p578]

M. D. Hoffman and A. Gelman. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623, 2014. [p586]

S. M. Iacus. *sde: Simulation and Inference for Stochastic Differential Equations*, 2016. URL https://CRAN.R-project.org/package=sde. R package version 2.0.15. [p583]

P. E. Jacob and S. Funk. *rbi: Interface to LibBi*, 2020. URL https://CRAN.R-project.org/package=rbi. R package version 0.10.3. [p578]

A. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970. [p581]

A. A. King, D. Nguyen, and E. L. Ionides. Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, 69(12):1–43, 2016. URL https://doi.org/10.18637/jss.v069.i12. [p578]

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996. [p580, 586]

F. Lindgren and H. Rue. Bayesian spatial modelling with R-INLA. *Journal of Statistical Software*, 63(19): 1–25, 2015. URL https://doi.org/10.18637/jss.v063.i19. [p578]

N. Michaud, P. de Valpine, D. Turek, C. Paciorek, and D. Nguyen. Sequential Monte Carlo methods in the nimble R package. Technical Report arxiv:1703.06206, arXiv preprint, 2020. [p578]

L. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Journal of Statistical Software, Articles*, 67(10):1–36, 2015. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v067.i10. [p578]

NIMBLE Development Team. nimbleSMC: Sequential Monte Carlo methods for NIMBLE, 2020. URL https://doi.org/10.5281/zenodo.1211190. R package version 0.10.0 https://cran.r-project.org/package=nimbleSMC. [p578]

G. Petris and S. Petrone. State space models in r. *Journal of Statistical Software*, 41(4):1–25, 2011. URL https://doi.org/10.18637/jss.v041.i04. [p578]

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006. URL https://CRAN.R-project.org/doc/Rnews/. [p584]

Stan Development Team. RStan: the R interface to Stan, 2020. URL http://mc-stan.org/. R package version 2.21.2. [p578]

F. Tusell. Kalman filtering in R. *Journal of Statistical Software*, 39(2):1–27, 2011. URL https://doi.org/10.18637/jss.v039.i02. [p578]

R. Van Der Merwe, A. Doucet, N. De Freitas, and E. A. Wan. The unscented particle filter. In *Advances in neural information processing systems*, pages 584–590, 2001. [p581, 586]

M. Vihola. Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22(5):997–1008, 2012. ISSN 1573-1375. URL `https://doi.org/10.1007/s11222-011-9269-5`. [p580]

M. Vihola, J. Helske, and J. Franks. Importance sampling type estimators based on approximate marginal MCMC. *Scandinavian Journal of Statistics*, 2020. URL `https://doi.org/10.1111/sjos.12492`. [p579, 581, 582, 586]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL `https://ggplot2.tidyverse.org`. [p584]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2021. URL `https://CRAN.R-project.org/package=dplyr`. R package version 1.0.7. [p584]

Jouni Helske
Department of Mathematics and Statistics
University of Jyväskylä
Finland
ORCiD: 0000-0001-7130-793X
`jouni.helske@jyu.fi`

Matti Vihola
Department of Mathematics and Statistics
University of Jyväskylä
Finland
ORCiD: 0000-0002-8041-7222
`matti.s.vihola@jyu.fi`

openSkies - Integration of Aviation Data into the R Ecosystem

by Rafael Ayala, Daniel Ayala, Lara Sellés Vidal, and David Ruiz

Abstract Aviation data has become increasingly more accessible to the public thanks to the adoption of technologies such as Automatic Dependent Surveillance-Broadcast (ADS-B) and Mode S, which provide aircraft information over publicly accessible radio channels. Furthermore, the OpenSky Network provides multiple public resources to access such air traffic data from a large network of ADS-B receivers. Here, we present **openSkies**, the first R package for processing public air traffic data. The package provides an interface to the OpenSky Network resources, standardized data structures to represent the different entities involved in air traffic data, and functionalities to analyze and visualize such data. Furthermore, the portability of the implemented data structures makes **openSkies** easily reusable by other packages, therefore laying the foundation of aviation data engineering in R.

Introduction

The ADS-B aircraft surveillance technology enables the tracking of aircraft through the reception by ground stations of a signal broadcast by the aircraft itself (Rekkas and Rees, 2008). Such a system does not require any external input and is instead fully dependent on data obtained by the aircraft navigation system. Additionally, unlike other radar-based systems used for aircraft surveillance, no interrogation signals from the ground stations are required. This is in opposition to, for example, the air traffic control radar beacon system operating in Mode S (Selective), where specific aircraft send a response message only after reception of another signal (the interrogation) emitted by a ground antenna. In the ADS-B, messages broadcast by an aircraft can include precise location information, typically determined by a Global Navigation Satellite System, such as GPS (Global Positioning System). Alternatively, it is possible to estimate the position of the aircraft from the timestamps of reception of the broadcast signal at different stations through multilateration (Kaune et al., 2012).

The OpenSky Network initiative provides free access to high-quality air traffic data derived from a network of thousands of ground stations that receive ADS-B messages, and, more recently, FLARM ("Flight Alarm") messages as well (Schäfer et al., 2014). Both ADS-B and FLARM messages are essentially radio signals broadcasted by an aircraft that can be received by other aircraft or ground stations. They carry positional information and other data about the broadcasting aircraft (with the main difference being that the FLARM system is primarily aimed at avoiding collisions. Therefore, it also transmits a prediction of the trajectory of the aircraft in the next 20 seconds). More than 25 trillion ADS-B messages and over 3 billion FLARM messages have been collected by the OpenSky Network since 2013, corresponding to over 400,000 aircraft in 190 countries. Such data provides the basis to develop novel algorithms for the determination of aircraft position and the analysis of air traffic data. Access to the available data is provided through two main tools: a live API (which also supports the retrieval of historical data, although with some limitations) and an Impala shell (Apache Software Foundation, 2020) that provides fast access to the full historical dataset, to which free access can be obtained for research and other non-profit purposes. However, these resources by themselves do not provide the data in appropriate data structures that make them amenable to advanced analyses available in R and other R packages.

We present **openSkies**, an R package that aims to provide a well-established and documented basis for the analysis of air traffic in R and to facilitate the future development of related algorithms. To that extent, access to the OpenSky Network live API and Impala shell is provided, as well as a decoder of raw ADS-B messages. A set of data structures that represent relevant concepts (such as aircraft, flights, or airports) is also implemented and used to store the data retrieved from the OpenSky Network. Relevant methods to process and analyze the corresponding data are associated with each data structure.

The package also includes functions to perform statistical analysis and to visualize the retrieved data, enabling, for example, the clustering of trajectories and the representation of aircraft and flight paths in a given airspace. Additional functionalities can be easily built on the already implemented classes and functions, either in future versions of **openSkies** or in other packages developed by the R community. In the following sections, we describe the different features of the package and demonstrate how each of them can be accessed and applied to real data.

Firstly, we describe the implemented data structures required to model air traffic (Data structures). We then show the functions that allow instantiating said classes with data obtained from the OpenSky Network (Information retrieval). Next, the currently available tools to visualize (Visualization of air traffic) and cluster (Clustering of aircraft trajectories) are presented. Finally, a tool to decode raw ADS-B messages is presented (Decoding ADS-B messages). Examples with real data are provided throughout the different sections to illustrate the features of the package.

Data structures

In order to establish standardized data structures that could serve as the basis for future developments in air traffic analysis in R, a set of R6 classes representing the frequently involved entities was implemented. R6 was chosen as the class system due to the possibility to establish formal definitions of reference classes, as well as the smaller memory footprint and faster speeds of object instantiation, field access, and field setting compared to base R's Reference Classes system.

Additionally, R6 classes are portable. Therefore, the classes defined in **openSkies** can be used by other packages.

In its current version (1.1.3), **openSkies** defines classes for the following entities: aircraft, airports, flight instances, flight routes, single aircraft state vectors, and series of aircraft state vectors (Figure 1).

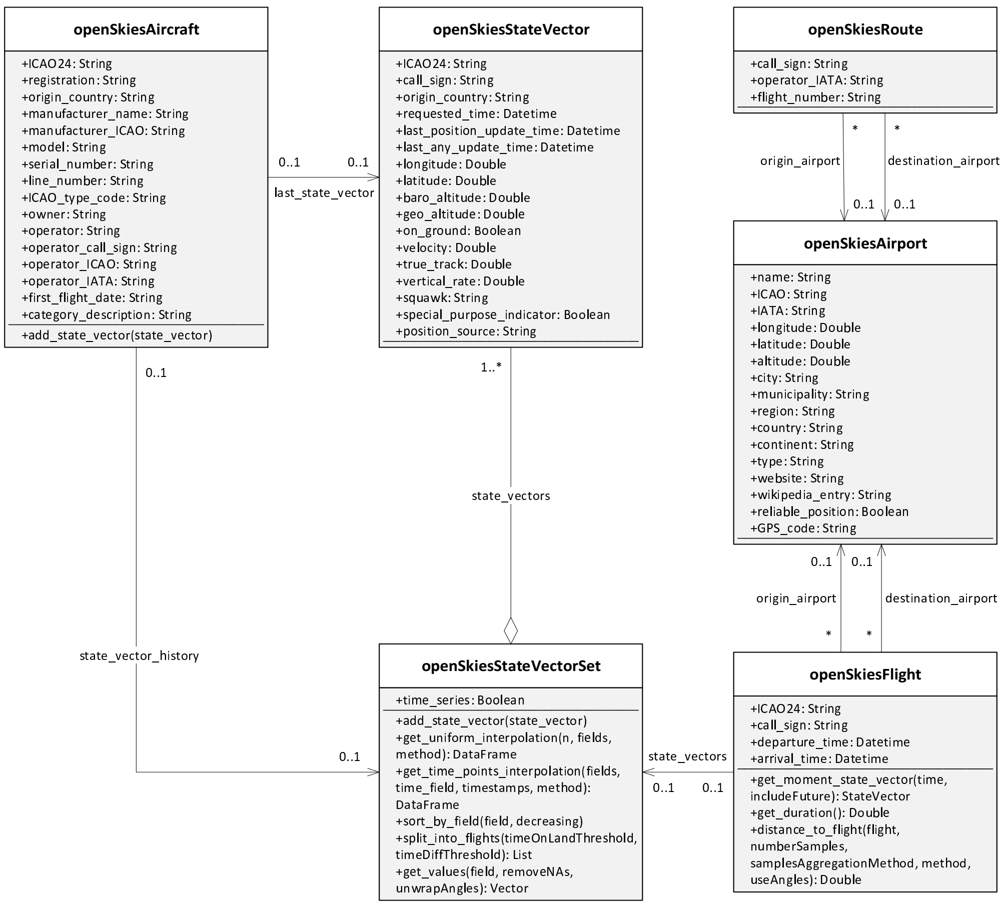


Figure 1: R6 classes implemented in **openSkies**.

Class "openSkiesStateVector" represents the state vector of an aircraft at a given time. A state vector comprises multiple pieces of information about the aircraft and its status, including positional information (latitude, longitude, altitude, whether the aircraft is on the ground) and trajectory information (velocity, track angle, vertical rate). Sets of "openSkiesStateVector" are represented with class "openSkiesStateVectorSet", which contains two fields: a list of objects of class "openSkiesStateVector" and a logical, indicating if the set of state vectors corresponds to a time series of the same aircraft (field `time_series`). Class "openSkiesStateVectorSet" contains methods to add more objects of class "openSkiesStateVector" to the contained list of state vectors, get interpolations of any of the numerical fields of the state vectors (uniformly spaced or at specific time points), sort the list of state vectors by a specific field, get the values of a given field for all the state vectors, and split the set of state vectors into individual flight instances. The latter operation is performed by firstly grouping the state vectors by aircraft and then splitting them into flights based on two conditions:

either the aircraft staying on the ground for a given amount of time or the aircraft not sending any status update for a given period (with default values of 300 seconds and 1800 seconds).

A flight instance is defined as a single flight performed by an aircraft, including typically (but not necessarily) take-off, air transit, and landing. Flights are represented class "openSkiesFlight", which contain fields for the aircraft that performed the flight, the departure and arrival times, the airports of origin and destination (as objects of class "openSkiesAirport"), and a set of state vectors describing the aircraft during the flight, which is an object of class "openSkiesStateVectorSet" with field time_series=TRUE. The class contains methods to get the state vector of the aircraft for a specified time, calculate the duration of the flight, and calculate the distance (based on features extracted from the trajectories) to another object of class "openSkiesFlight".

Airports are represented by class "openSkiesAirport", with fields describing, among others, the airport codes, its position (latitude, longitude, and altitude), and its location in terms of administrative divisions of different levels.

Flight routes, which differ from flight instances in that they are designated paths followed regularly by aircraft to go from one airport to another one, are represented by class "openSkiesRoute". The class contains fields for the call sign and flight number associated with the route, its operator, and its airports of origin and destination (objects of class "openSkiesAirport").

Finally, aircraft are represented by class "openSkiesAircraft", which contains fields describing the aircraft model, registration, manufacturer, owner, and operator. Objects of this class can also contain a field with an object of class "openSkiesStateVectorSet" representing the history of known state vectors of the aircraft in a given period.

Information retrieval

While it is possible to manually instantiate any of the classes previously described by providing values for the required fields with information obtained from any source, a more convenient way is to instantiate them from information automatically retrieved from the OpenSky Network. To that extent, a series of functions that retrieve data from the different resources offered by the OpenSky Network is implemented. State vectors can be accessed with two functions. Firstly, getSingleTimeStateVectors retrieves all the state vectors received at a specified time as an "openSkiesStateVectorSet" object with field time_series=FALSE. In the default behavior, state vectors received from any aircraft at any location are returned. However, the results can be filtered by specifying one or more aircraft and the boundaries of an area of interest. On the other hand, getAircraftStateVectorsSeries retrieves a time series of state vectors for a given aircraft with the specified time resolution, in the form of an "openSkiesStateVectorSet" object with field time_series=TRUE. Both functions can retrieve data from the OpenSky Network through two resources integrated seamlessly. By default, the live API is employed. The live API does not require registration but imposes limitations (larger for anonymous users) on the retrieval of historical data and the time resolution with which data can be accessed. Additionally, large queries can take considerable amounts of time, especially when retrieving time series of state vectors. Therefore, the functions can also access the database through the Impala shell, which requires registration but does not impose such limits and performs queries significantly faster. Authentication for these two functions can be performed by providing login details through the username and password arguments, and access through the Impala shell can be enabled by authorized users simply by setting argument useImpalaShell=TRUE. New users can register through the OpenSky Network website, and Impala shell access should be directly requested to the OpenSky Network administrators.

In both cases, the resulting objects of class "openSkiesStateVectorSet" can be used to instantiate objects of class "openSkiesFlight" by calling the split_into_flights method. Alternatively, functions to retrieve flight instances from the OpenSky Network are available, based on the aircraft that performed them (getAircraftFlights) or the airport of origin (getAirportDepartures) or destination (getAirportArrivals). In all cases, it is possible to include the time series of state vectors corresponding to each flight by setting includeStateVectors=TRUE and airport metadata through includeAirportsMetadata=TRUE.

Airport and aircraft metadata can also be retrieved independently with the getAirportMetadata and getAircraftMetadata functions by providing the ICAO 4-letter code of the airport of interest or the ICAO 24 bit address of the target aircraft, respectively.

Finally, information about routes can be accessed by providing the call sign of the route to the getRouteMetadata function, which includes the possibility to retrieve the metadata of the associated airports of origin and destination if includeAirportsMetadata=TRUE.

In the following example, we aim to demonstrate some of the most common methods for retrieving air traffic data and analyzing it.

```
library(openSkies)

## In the following example, we retrieve information about all flights
## that departed from Frankfurt Airport the 29th of January 2018 after 12 pm.
## It should be noted that such large requests can take a long time unless
## performed with the Impala shell
## It should also be noted that, in this and the following examples, the
## values for the username and password arguments should be substituted with
## personal credentials registered at the OpenSky Network
flights <- getAirportArrivals(airport="EDDF", startTime="2018-01-29 12:00:00",
                              endTime="2018-01-29 24:00:00", timeZone="Europe/Berlin",
                              includeStateVectors = TRUE, timeResolution = 60,
                              useImpalaShell = TRUE, username = "user",
                              password = "password")
## We can then easily check the amount of flights
length(flights) # 316 flights

## Trajectories of the 316 flights can be obtained by retrieving
## the set of state vectors of each flight
trajectories_frankfurt <- lapply(flights, function(f) f$state_vectors)

## It is also possible to retrieve all state vectors received from a
## given aircraft in a given period of time. In the following example, we
## obtain all the state vectors received for aircraft with ICAO24 code
## 403003 between 12 PM of the 8th of October, 2020 and 6 PM of the
## 9th of October, 2020
stateVectors <- getAircraftStateVectorsSeries("403003", startTime = "2020-10-08 12:00:00",
                                              endTime = "2020-10-09 18:00:00",
                                              timeZone="Europe/London",
                                              timeResolution = 60,
                                              useImpalaShell = TRUE,
                                              username = "user",
                                              password = "password")

## The ensemble of state vectors can then be split into individual
## flight instances, revealing that the aircraft performed 6 flights
## in the analyzed period
flights <- stateVectors$split_into_flights()
length(flights) # 6 flights

## Let us get some additional information about the flights performed by
## this aircraft. For example, the maximum speed that it reached in km/h
maxSpeed <- max(stateVectors$get_values("velocity", removeNAs = TRUE))/1000*3600

## The maximum speed that it reached is just above 210 km/h. This is well below
## the speed typically used by commercial passenger jets, usually around 800 km/h
## Investigation of the aircraft model confirms that it is indeed a small CESSNA 152
aircraftData <- getAircraftMetadata("403003")
aircraftData$model
```

Visualization of air traffic

Two main methods for visualization of air traffic are currently available in **openSkies**: plotting of flights and plotting of aircraft at a given time.

The first method aims to represent flights performed by one or more aircraft over a given period of time. If only a single flight is to be plotted, the `plotRoute` function can be used, which receives as its main input an object of class `"openSkiesStateVectorSet"` with `time_series=TRUE`. Alternatively, `plotRoutes` can be used for plotting multiple flights, receiving in this case as input a list of `"openSkiesStateVectorSet"` objects with `time_series=TRUE`. The color of the routes can be manually set by providing a vector of color names to `pathColors` and setting `literalColors=TRUE`. It is also possible to color the routes according to the value of any property by providing such values as a factor to `pathColors` and setting `literalColors=FALSE`.

The second visualization method (plotPlanes) enables the plotting of all aircraft flying over a defined area at a particular time. An "openSkiesStateVectorSet" object with each state vector representing the position of an aircraft at the desired time should be provided as input, most frequently obtained through getSingleTimeStateVectors.

All plotting functions allow plotting onto any object of class ggmap from the **ggmap** package (Kahle et al., 2019). If no ggmap object is provided, one will be created internally, with boundaries large enough to contain all the positions in the provided "openSkiesStateVectorSet" objects, plus additional space determined by the paddingFactor argument.

The following examples demonstrate the application of visualization functions. First, the formerly retrieved trajectories for the CESSNA 152 are plotted with different colors:

```
## First, let us obtain the trajectories of the flights performed
## by the CESSNA 152
trajectories_CESNA152 <- lapply(flights, function(f) f$state_vectors)

## Then, we create a color palette
library(viridis)
colors <- magma(length(trajectories))

## Then, the trajectories are plotted
plotRoutes(trajectories, pathColors = colors, lineSize = 1.2,
           lineAlpha = 0.8, includeArrows = TRUE,
           paddingFactor = 0.05)
```

The resulting plot is shown in Figure 2. Next, we plot all the aircraft flying over Switzerland in a given instant:

```
## Firstly we retrieve the state vectors of all aircraft flying over Switzerland
## the 2nd of March, 2018 at 14.30 (London time)
vectors_switzerland <- getSingleTimeStateVectors(time="2018-03-02 14:30:00",
                                                 timeZone="Europe/London",
                                                 minLatitude=45.8389,
                                                 maxLatitude=47.8229,
                                                 minLongitude=5.9962,
                                                 maxLongitude=10.5226,
                                                 username="user",
                                                 password="password",
                                                 useImpalaShell = TRUE)

## Then, the aircraft are plotted
plotPlanes(vectors_switzerland)
```

The result is shown in Figure 3.

Clustering of aircraft trajectories

As a means to analyze trajectories, **openSkies** provides the functionalities to perform clustering of trajectories. The first step is to obtain a matrix of features of the trajectories to be clustered. This can be achieved by providing a list of "openSkiesStateVectorSet" objects to the getVectorSetListFeatures function.

The "openSkiesStateVectorSet" objects in the list should each correspond to a single flight instance, and therefore it is advisable to first split the data into individual flights with method split_into_flights. It is possible to include track angles in the generated features matrix by setting useAngles=TRUE, which can be desirable if flights going along the same route but in opposite directions should be classified separately.

The computed features matrix (or, alternatively, the list of "openSkiesStateVectorSet" objects, in which case the features matrix will be calculated internally) can then be passed to the clusterRoutes function. A number of different clustering algorithms can be selected through the method argument. Possible methods are dbscan, kmeans, hclust, fanny, clara, and agnes.

The desired number of clusters can be specified through numberClusters, although this argument is ignored if the DBSCAN algorithm (Hahsler and Piekenbrock, 2019) is used since it does not require *a priori* determination of the number of clusters. Instead, the *eps* parameter controls the size of the

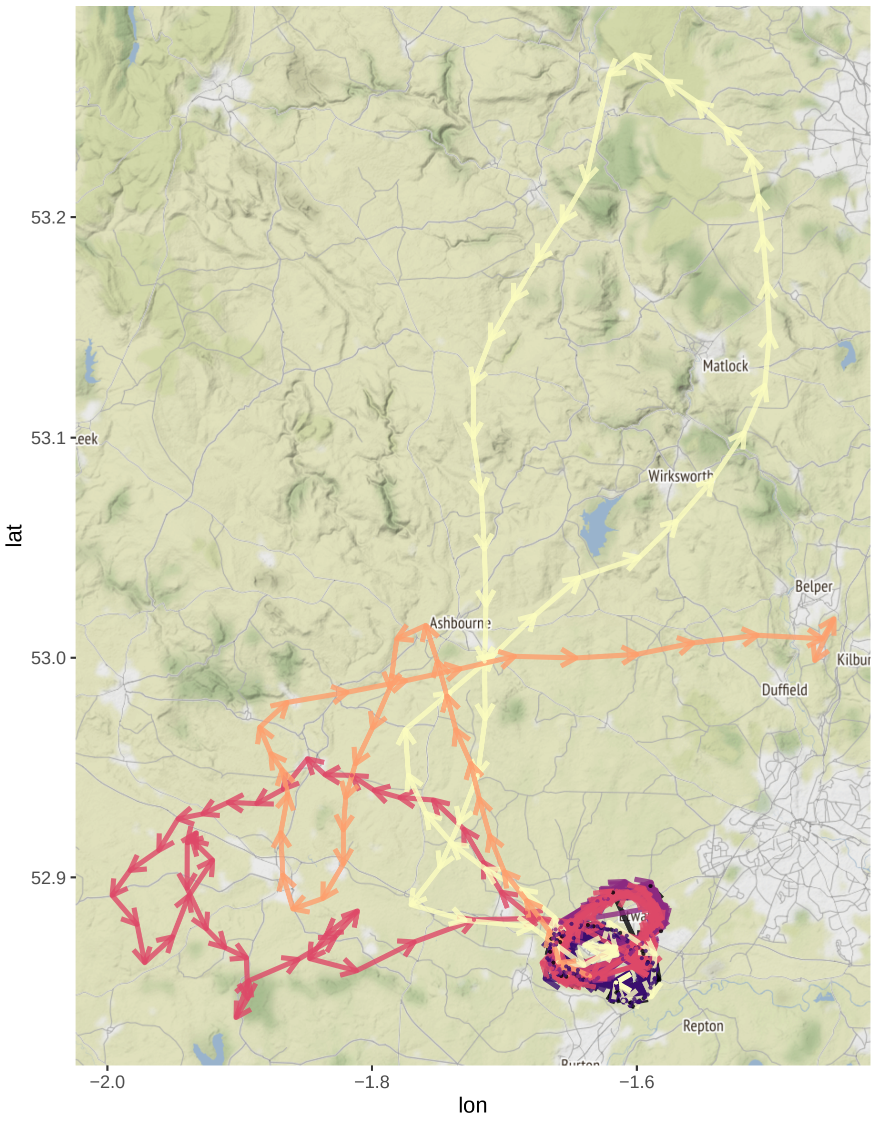


Figure 2: Visualization of flights performed by a CESSNA 152.

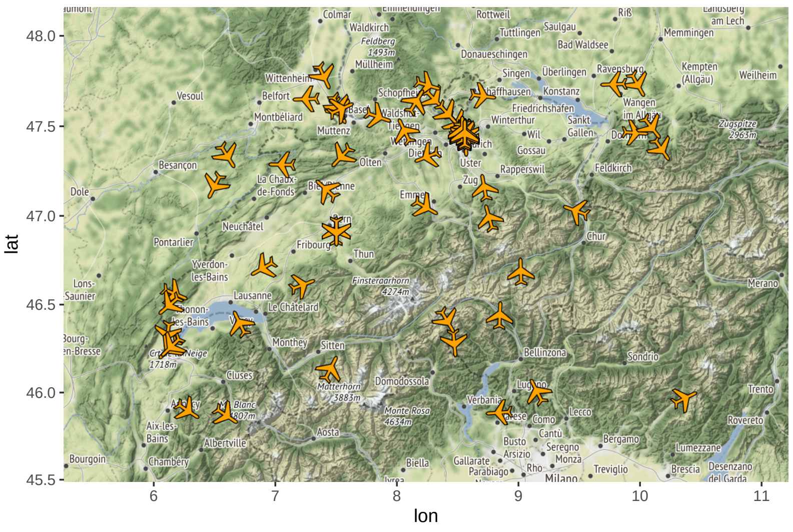


Figure 3: Visualization of aircraft flying over Switzerland.

epsilon neighborhood to be passed to the DBSCAN, which influences the number of found clusters (any two trajectories will be considered to belong to the same cluster if their calculated distance is below the threshold epsilon).

For all other methods (Maechler et al., 2019), if a number of clusters is not chosen, it will be internally estimated. The results of clustering can be visualized by providing the factor with the assigned clusters to the pathColors argument of plotRoutes. In the following example, we perform clustering of the previously obtained set of 316 flights departing from Frankfurt airport (Figure 4).

```
## As an example, let´s cluster the previously retrieved 316 flights departing
## from Frankfurt airport into 8 clusters with the k-means algorithm
clusters = clusterRoutes(trajectories_frankfurt, "kmeans", numberClusters = 8)

## The results can be visualized with plotRoutes
plotRoutes(trajectories_frankfurt, pathColors = clusters$cluster,
           literalColors = FALSE, paddingFactor = 0.1)

## The clusters differ in their broad area of destination. For example,
## clusters 4, 5 and 6 comprise mostly flights to North America, Japan and
## central Asia respectively. Due to their geographical proximity, it is expected
## that flights from cluster 5 are closer to flights from cluster 6 than
## to those assigned to cluster 4. Let us confirm this:
flights[clusters$cluster==5][[1]]$distance_to_flight(flights[clusters$cluster==6][[1]])
## Returns 123.3449
flights[clusters$cluster==5][[1]]$distance_to_flight(flights[clusters$cluster==4][[1]])
## Returns 348.4957

## The shorter distance between flights of clusters 5 and 6 is in accordance with
## the expectations.
```

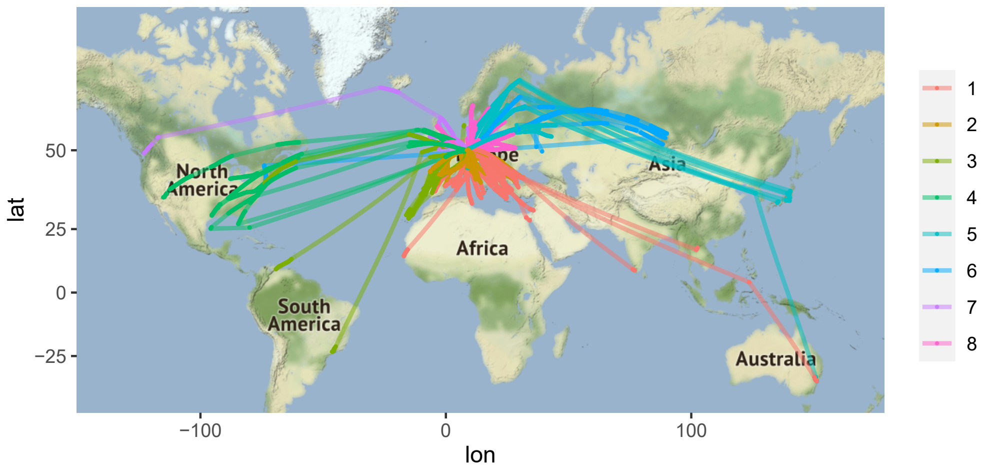


Figure 4: Clustering of flights departing from Frankfurt airport.

Decoding ADS-B messages

Finally, **openSkies** also provides a decoder of raw ADS-B messages, implemented as an instance of R6 class "adsbDecoder" with the methods required to decode messages. This tool is useful for users willing to extract air traffic data from their own sources, such as private ADS-B receivers. Firstly, the messages, typically available in hexadecimal format, must be converted to binary format, which can be achieved with the hexToBits function. The binary messages can then be passed to the decodeMessage method, which decodes a single message. It should be noticed that, due to the Compact Position Reporting format with which the position of airborne aircraft is encoded, at least 2 consecutive positional ADS-B messages are required in order to unequivocally determine the position of the aircraft (referred to as *even* and *odd* frames).

Therefore, the decoder caches the most recently even and odd decoded message, to decode the next complementary positional message.

It is also possible to provide a list of messages in binary format to the decodeMessages method, which will directly provide positional information about the aircraft if the list contains two complementary even and odd positional messages. In the following example, we demonstrate both the sequential and batch decoding of ADS-B messages:

```
## Let use the following set of 2 ADS-B messages in hexadecimal format
## to illustrate the ADS-B decoder
message1 <- "8D40621D58C386435CC412692AD6"
message2 <- "8D40621D58C382D690C8AC2863A7"

## First, the messages must be converted to binary format:
message1_bin <- ADSBDecoder$hexToBits(message1)
message2_bin <- ADSBDecoder$hexToBits(message2)

## We can then obtain positional information by sequentially decoding
## both messages. Note that no positional information is obtained from the
## data extracted from message1, since this is the first message of the two
## that, as a pair, encode positional information.
message1_decoded <- ADSBDecoder$decodeMessage(message1_bin)
message2_decoded <- ADSBDecoder$decodeMessage(message2_bin)
message2_decoded$data$lat # The latitude is 52.2572 degrees North
message2_decoded$data$lon # The longitude is 3.919373 degrees East

## We can also provide both messages as a list to decode them simultaneously:
all_messages_decoded <- ADSBDecoder$decodeMessages(list(message1_bin, message2_bin))
```

Future work

openSkies is in active development. Therefore, novel features are expected to be available through frequent updates.

Some of the features that will become available in the short and medium-term include:

- Smoothing and filtering of trajectories to remove spurious position outliers through both linear and non-linear methods.
- Automatic detection of the different segments of flight instances (such as take-off, initial climb, cruise altitude, descent, and landing).
- Detection of in-flight events and flight patterns. For example, delays in landing due to multiple reasons, which are often seen as small periodical trajectories near the destination airport.
- Application of detected in-flight events and flight patterns to obtain additional information about the involved aircraft, such as aircraft, type when this is not readily available.

Summary

New technologies such as ADS-B surveillance and initiatives like the OpenSky Network have largely increased the amount of air traffic data that is publicly available, as well as its accessibility. However, the data provided by said resources must be parsed and formatted adequately in order to be analyzed in-depth and serve as the starting point to develop new algorithms useful for the field, such as novel multilateration methods.

However, while appropriate toolboxes have been implemented in other languages (Olive and Basora, 2019), such a standardized set of tools was not available in R until the development of **openSkies**, which aims to fill this void. One of the key features of our package is the automation and streamlining of the data engineering required in order to access air traffic data at a large scale and to enable data scientists to make meaningful analyses with it. The implemented portable data structures and functionalities provide a solid base to apply the vast range of functionalities provided by other R packages to air traffic data. The ecosystem created with future packages that interoperate with **openSkies**, together with reference datasets (Schäfer et al., 2020), should serve to standardize and expand the scope of air traffic data analysis in R.

Acknowledgements

The development of this work is supported by the Spanish Ministry of Science and Innovation (grant code PID2019-105471RB-I00) and the Regional Government of Andalusia (grant code P18-RT-1060).

Bibliography

Apache Software Foundation. *Apache Impala Guide*, 2020. URL `http://impala.apache.org/docs/build/impala-3.4.pdf`. [p590]

M. Hahsler and M. Piekenbrock. *dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms*, 2019. URL `https://CRAN.R-project.org/package=dbscan`. R package version 1.1-5. [p594]

D. Kahle, H. Wickham, and S. Jackson. *ggmap: Spatial Visualization with ggplot2*, 2019. URL `https://CRAN.R-project.org/package=ggmap`. R package version 3.0.0. [p594]

R. Kaune, C. Steffes, S. Rau, W. Konle, and J. Pagel. Wide area multilateration using ADS-B transponder signals. In *2012 15th International Conference on Information Fusion*, pages 727–734, 2012. URL `https://ieeexplore.ieee.org/document/6289874`. [p590]

M. Maechler, P. Rousseeuw, A. Struyf, and M. Hubert. *cluster: "Finding Groups in Data": Cluster Analysis Extended*, 2019. URL `https://CRAN.R-project.org/package=cluster`. R package version 2.1.0. [p596]

X. Olive and L. Basora. A Python Toolbox for Processing Air Traffic Data: A Use Case with Trajectory Clustering. In *Proceedings of the 7th OpenSky Workshop 2019*, pages 73–84, 2019. URL `https://doi.org/10.29007/sf1f`. [p597]

C. Rekkas and M. Rees. Towards ADS-B implementation in Europe. In *2008 Tyrrhenian International Workshop on Digital Communications - Enhanced Surveillance of Aircraft and Vehicles*, pages 1–4, 2008. URL `https://doi.org/10.1109/TIWDC.2008.4649019`. [p590]

M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm. Bringing up OpenSky: A large-scale ADS-B sensor network for research. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pages 83–94, 2014. URL `https://doi.org/10.1109/IPSN.2014.6846743`. [p590]

M. Schäfer, M. Strohmeier, M. Leonardi, and V. Lenders. LocaRDS: A Localization Reference Data Set, 2020. URL `https://arxiv.org/abs/2012.00116`. *arXiv* preprint arXiv:2012.00116. [p597]

Rafael Ayala
Okinawa Institute of Science and Technology Graduate University
7542 Onna, Onna-Son, Kunigami, Okinawa 904-0411
Japan
ORCiD: 0000-0002-9332-4623
`rafael.ayala@oist.jp`

Daniel Ayala
University of Seville
ETSI Informática, Avda. de la Reina Mercedes, s/n, Sevilla E-41012
Spain
ORCiD: 0000-0003-2095-1009
`dayala1@us.es`

Lara Sellés Vidal
Imperial College London
Exhibition Road, London, SW7 2AZ
United Kingdom
ORCiD: 0000-0003-2537-6824
`lara.selles12@imperial.ac.uk`

David Ruiz
University of Seville

ETSI Informática, Avda. de la Reina Mercedes, s/n, Sevilla E-41012
Spain
ORCiD: 0000-0003-4460-5493
druiz@us.es

Generalized Linear Randomized Response Modeling using GLMMRR

by Jean-Paul Fox, Konrad Klotzke and Duco Veen

Abstract Randomized response (RR) designs are used to collect response data about sensitive behaviors (e.g., criminal behavior, sexual desires). The modeling of RR data is more complex since it requires a description of the RR process. For the class of generalized linear mixed models (GLMMs), the RR process can be represented by an adjusted link function, which relates the expected RR to the linear predictor for most common RR designs. The package **GLMMRR** includes modified link functions for four different cumulative distributions (i.e., logistic, cumulative normal, Gumbel, Cauchy) for GLMs and GLMMs, where the package **lme4** facilitates ML and REML estimation. The mixed modeling framework in **GLMMRR** can be used to jointly analyze data collected under different designs (e.g., dual questioning, multilevel, mixed mode, repeated measurements designs, multiple-group designs). Model-fit tests, tools for residual analyses, and plot functions to give support to a profound RR data analysis are added to the well-known features of the GLM and GLMM software (package **lme4**). Data of Höglinger and Jann (2018) and Höglinger, Jann, and Diekmann (2014) are used to illustrate the methodology and software.

Introduction

Response distortion is known to be a potential threat to accurate (survey) results. The methodological literature shows many examples of misreporting of embarrassing or socially undesirable behaviors in surveys (Tourangeau and Yan, 2007; Tourangeau and Smith, 1996). The randomized response technique (RRT) has been developed to encourage respondents to answer truthfully to questions about sensitive behaviors. The RRT is designed to collect responses in an indirect way, where respondents are instructed to answer a sensitive question truthfully only with a certain probability. Therefore, an affirmative response to a sensitive question is masked by a random device at the individual level. The traditional way of direct questioning (DQ) about sensitive topics is known to lead to socially desirable responses or non-cooperation of the respondents. Honest responding is encouraged by RRT by increasing anonymity and confidentiality.

After the introduction of the randomized response (RR) design by Warner (1965), many randomized response (RR) designs have been introduced to improve RRT performance in collecting sensitive information. Warner's design is based on a deck of cards containing cards with the sensitive question and cards with its negation. Since then, different randomizing devices have been introduced to improve the privacy protection of the respondent. Outcomes of rolling dice or spinning a spinner have been used to determine at random one of the possible routings to response for a respondent. Another non-sensitive question has been introduced to make the design easier to implement, where the response to the non-sensitive question is used to steer the response process at random. For online surveys, an online randomizing device, such as a digital dice or spinner, is less reliable since the computer outcomes can be stored. Therefore, techniques have been developed where user interaction is requested to generate a random outcome, which cannot be digitally stored. Currently, there are many ways to randomize the response in an online survey, where the randomizing design properties – the level of privacy protection and the efficiency of the design – are still under the control of the investigator.

The flexibility in RR designs has also improved the utility of the RRT. They can be applied to survey items (Lensvelt-Mulders et al., 2005; van den Hout et al., 2010) but also to scale items for measuring sensitive constructs (Fox, 2005; Fox and Meijer, 2008; Fox et al., 2013). Different RRT implementations have been introduced and applied in many research fields. Together with this expansion of the RRT to different domains, statistical tools have been developed to improve the analysis of RR data. Scheers and Dayton (1988) developed a (logistic) linear regression method for RR data to explore relationships between variables in addition to estimating prevalence rates of the sensitive attribute. Since then, more advanced statistical regression models for RR data have been developed. For instance, randomized item-response theory (RIRT) models for measuring sensitive constructs (Fox, 2012) and mixture RR models to deal with non-cooperating participants (Fox et al., 2013). To further stimulate the use of RRT, software tools are needed to make the developed methods and designs for RR data widely available.

Recently, two packages for RR data have become available in R. The package **rr** of Blair et al. (2015) was developed for univariate power analysis to measure the sensitive item prevalence for four RR designs. It also comprehends a logistic regression function for RR data collected with a single RR design. The package **RRreg** of Heck and Moshagen (2018) has logistic and linear regression routines, which can include random effects, and tools for power analysis for different RR designs (including those for continuous data). However, both packages are restricted to single-group RR designs and do not allow joint analysis of different randomization schemes across items and/or participants.

Multiple-group RR designs are particularly interesting in online surveys, where different RR designs and different random devices can be easily varied. For instance, the level of protection can be moderated by varying the design parameters, and RR designs can be varied across survey items to improve truth-telling. For validation studies, multiple-group RR designs are relevant to make comparisons between groups questioned with different RR designs (Höglinger and Jann, 2018; Höglinger et al., 2016). For a meta-analysis, a joint analysis of the available data can improve the measurement of a sensitive prevalence. Furthermore, differences in prevalence estimates can be examined across studies. More recently, there has been an increased interest in measuring the efficacy of the RRT to improve the honest disclosure of sensitive information (John et al., 2018). This is usually done through validation studies where a DQ group serves as the baseline, and it is investigated whether higher prevalence estimates are obtained with a single-group RRT. The huge amount of literature on the topic and mixed results about the benefits of RRT has led to skeptical positions toward RRT (Wolter and Preisendörfer, 2013). However, the RR design parameters (the random-device properties), as well as the type of RR design, can influence the response behavior, and care must be taken in designing the RRT to reduce the level of misreporting. It is not likely that a fixed privacy level encourages truth-telling in a uniform way across participants. The performance of RRT will vary across subgroups and participants, and different RR designs can vary in performance across subgroups depending on the level of sensitivity of the question.

The R package **GLMMRR** extends existing implementations by providing generalized regression tools for multiple-group RR designs. It builds in a natural way on common regression methods of the system package **stats** to make them suitable for RR data. The R package **GLMMRR** gives support to *generalized linear (mixed effects) regression modeling* of binary RR data and extends the popular generalized linear regression routines in R to handle RR outcomes by including RR-specific link functions for a wide variety of RR designs. The **GLMMRR** provides several important contributions for the joint regression analysis of binary RR data:

1. Different link functions (logit, probit, complementary log-log, cauchit) are supported to optimize the link between the linear predictor and the response and to avoid restrictions on the range of the (expected) response. The complementary log-log and cauchit link functions approach the asymptotes of zero and one asymmetrically.
2. Each link function is modified to an RR-link function to make it suitable for various RR designs (Warner, Unrelated question, Forced response, Kuk, Crosswise, Triangular), where two design parameters specify the properties of the RR design.
3. The package supports a joint regression analysis of RR data sampled with different RR designs and different design parameters. Each response observation is either indirectly observed with an RR design with unique design parameters or directly observed without an RR design.
4. For randomized item-response data, RR-design specific (weighted) prevalence rates and confidence intervals can be computed for each item. Trait levels can be estimated, where the items serve as indicators of a sensitive trait.
5. Extensive residual tools are available for the evaluation of the fit of the model. This includes Pearson, deviance and response residuals, goodness-of-fit statistics (Pearson's and deviance chi-square statistic, Hosmer-Lemeshow statistic), and the Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC).
6. The `RRglm` is used to fit a GLM, and an object of class `"RRglm"` is created, which extends the class `"glm"` with RR data. The `glm` methods are applicable to an object of class `"RRglm"`. The function `summary` or `print` can be used to obtain a summary of the results. Other useful features can be extracted using the well-known generic functions, such as `coefficients`, `effects`, `fitted.values`, and `anova`.
7. The `RRglmer` for fitting a GLMM returns an object of class `"RRglmerMod"` which extends the class `"glmerMod"` and `"merMod"` with RR data. The methods for an object of class `"glmerMod"` are applicable for an object of class `"RRglmerMod"` (the methods can be found in the **lme4** documentation).

This paper is organized as follows. In the next section, the RR-link functions are introduced, and the GLM and GLMM modeling framework for RR data is introduced. A general introduction is given where the responses are distributed according to a distribution of the exponential family, and the RR-link function is used to link the linear predictor to the expected (randomized) response. In Section 2.3, the ML and REML estimation method is discussed for the GLM(M) with RR-link functions. Then, the RIRT modeling approach for measuring sensitive constructs is discussed.

Section 2.5 discusses different methods to evaluate the fit of the model. Different residuals are defined (Pearson, deviance, response) for a GLM. For a GLMM, a conditional residual is defined as the difference between the observation and the conditional expected value (given the random effect estimate). Goodness-of-fit statistics are introduced by grouping the Pearson and deviance residuals. For a fixed number of groups, the Hosmer-Lemeshow (H-L), the Pearson, and the deviance statistic are approximately chi-square distributed and can be used to evaluate the global fit of a GLM. In Section 2.6, the two main functions of the package are discussed for fitting GLMs and GLMMs with RR data. In Section 2.7, an illustration of the package is shown by analyzing data from the validation study of Höglinger and Jann (2018), where differences in outcomes between RR designs are evaluated. Furthermore, the prevalence of student misconduct is analyzed with a randomized item-response theory model where students were assigned to different treatment groups, each assigned to a specific RR design. A joint modeling approach is carried out to examine differences between treatment groups. Then, in Section 2.8, the conclusions are given.

General randomized-response probability model

When collecting RR data, each observation is randomized before it is observed. The RR process makes it impossible to directly relate an observed RR to the sensitive question. It masks the answers of respondents. To ease the notation, a distinction is made between respondents with index i and items with index k. A further specification is possible where participants and/or items are grouped, which will be shown in the real data examples in Section 2.7. Thus, respondent's i prevalence to the sensitive question k, denoted as $\tilde{\pi}_{ik}$, cannot be directly measured. Instead, the RR probability, denoted as π_{ik}, is measured.

Fox et al. (2018) showed that for the common RR designs for binary data, the RR probability for the observed RR data can be related to the prevalence of the sensitive question through a linear equation. This linear relationship is given by,

$$\pi_{ik} \quad = \quad c_{ik} + d_{ik}\tilde{\pi}_{ik}. \tag{1}$$

The RR parameters c and d determine the type of RR design. A variety of RR designs can be represented by Equation (1): Warner's design (Warner, 1965), Unrelated Question (UQM) design, Forced response (FR) design (Boruch, 1971), Kuk's design (Kuk, 1990), Triangular design (Yu et al., 2008), and the Crosswise (CW) design (Yu et al., 2008). The RR parameters c and d are retrieved from the RR design parameters. For instance, for Warner's design, a positive response is given to the sensitive question with probability p_1 or a negative response to its negation with probability $1 - p_1$,

$$\begin{aligned} \pi_{ik} &= p_1\tilde{\pi}_{ik} + (1-p_1)(1-\tilde{\pi}_{ik}) \\ &= \underbrace{(1-p_1)}_{c_{ik}} + \underbrace{(2p_1-1)}_{d_{ik}}\tilde{\pi}_{ik}. \end{aligned}$$

The parameters c_{ik} and d_{ik} describe the random response process, which are allowed to vary across questions and respondents such that the data-collection design is defined for every single response. Fox et al. (2018) give an overview in which the RR-design parameters are represented as RR parameters for various RRTs.

Although the RR designs work in different ways, a general RR model can be defined, which relates the RR data to the prevalence of the sensitive question. This general RR model is extended to a GLM and GLMM by linking the prevalence $\tilde{\pi}_{ik}$ to a linear predictor. The **GLMMRR** functions require the RR design parameters p_1 and p_2. The function `getRRparameters` provides the RR parameters c and d given the type of RR design and corresponding parameter values. This is illustrated in our applications in Section 2.7, where the RR designs UQM, FR, and CW are used, and the DQ design serves as a baseline.

Modified link functions for GLMs and GLMMs

The object is to define the natural link function, which relates the expected RR to a linear predictor for the prevalence $\tilde{\pi}_{ik}$. The RR parameters c_{ik} and d_{ik} are integrated into the link function to relate the expected RR observation to this linear predictor without explicitly parameterizing the prevalence. In this approach, the natural link functions for Bernoulli distributed observations are modified to account for the RR parameters. Simply by adjusting the natural link functions, the class of GLMs and GLMMs is extended to (binary) RR data.

Let Y_{ik} denote the (binary) RR observation of respondent i to item k. For binary data, the expected response is equal to the RR probability, π_{ik}. The expected response is related to a linear predictor denoted as η_{ik}, which is a linear combination of predictor variables for respondent i related to the response to item k. For binary RR data, the expected RR observation is equal to the RR probability, π_{ik}, and the relationship with the η_{ik} can be represented by

$$\begin{aligned} E\left(Y_{ik} \mid \eta_{ik}\right) &= \pi_{ik} \\ &= c_{ik} + d_{ik}\,\tilde{\pi}_{ik} \\ &= c_{ik} + d_{ik} g^{-1}\left(\eta_{ik}\right) \\ &= c_{ik} + d_{ik} g^{-1}\left(\mathbf{x}_{ik}^{t}\boldsymbol{\beta} + \mathbf{z}_{ik}^{t}\boldsymbol{b}_i\right), \end{aligned} \tag{2}$$

where the explanatory variables $\mathbf{x}_{ik}$ have fixed effects (i.e., common across respondents) and the $\mathbf{z}_{ik}^t$ have random effects (i.e., vary across respondents). The random effect $\boldsymbol{b}_i$ is assumed to have a multivariate normal distribution with mean zero and variance $\boldsymbol{\Sigma}_{\boldsymbol{b}}$. For $c_{ik} = 0$ and $d_{ik} = 1$, the $g^{-1}()$ is the mean function for (non-randomized) Bernoulli distributed observations. It follows that the natural link function for Bernoulli distributed data, $g()$, needs to be modified to link the expected RR to the linear predictor for the prevalence $\tilde{\pi}_{ik}$,

$$g\left(\frac{\pi_{ik} - c_{ik}}{d_{ik}}\right) = \eta_{ik}. \tag{3}$$

Consider the Bernoulli distributed RR observation with success probability π_{ik}, for which four different cumulative distribution functions can be applied (i.e., logistic, probit, Gumbel, Cauchy). Then, the distribution of the RR is given by

$$Y_{ik} \sim \mathcal{B}\left(\pi_{ik}\right) \tag{4}$$

$$\pi_{ik} = c_{ik} + d_{ik} g^{-1}\left(\eta_{ik}\right) \tag{5}$$

$$= \begin{cases} c_{ik} + d_{ik}\frac{\exp(\eta_{ik})}{1+\exp(\eta_{ik})} & \text{Logistic} \\ c_{ik} + d_{ik}\Phi\left(\eta_{ik}\right) & \text{Probit} \\ c_{ik} + d_{ik}\left(1 - \exp\left(1 - \exp\left(\eta_{ik}\right)\right)\right) & \text{Gumbel} \\ c_{ik} + d_{ik}\left(\arctan\left(\eta_{ik}\right)/\pi + \frac{1}{2}\right) & \text{Cauchy.} \end{cases} \tag{6}$$

Next, four different link functions are defined by modifying the linear predictor's relation with the prevalence, such that it relates to the expected RR observation. The possible (modified) link functions are given by,

$$\eta_{ik} = g\left(\left(\pi_{ik} - c_{ik}\right)/d_{ik}\right) \tag{7}$$

$$= \begin{cases} \ln\left(\frac{\pi_{ik}-c_{ik}}{c_{ik}+d_{ik}-\pi_{ik}}\right) & \text{Logit Link} \\ \Phi^{-1}\left(\frac{\pi_{ik}-c_{ik}}{d_{ik}}\right) & \text{Probit Link} \\ \ln\left(-\ln\left(\frac{c_{ik}+d_{ik}-\pi_{ik}}{d_{ik}}\right)\right) & \text{Complementary log-log Link} \\ \tan\left(\pi\left(\frac{\pi_{ik}-c_{ik}}{d_{ik}}\right)\right) & \text{Cauchit Link.} \end{cases} \tag{8}$$

The considered four different link functions are modified versions of the common link functions as defined in, for example, McCullagh and Nelder (1989) and Tutz (2011). When $c_{ik} = 0$ and $d_{ik} = 1$, the common link functions for directly observed responses are given.

Exponential family distributions

The modified (natural) link functions can be employed for (Bernoulli) exponential family distributed RR data. Assume the observed RR data is distributed according to a distribution of the exponential family; that is,

$$p\left(y_{ik} \mid \theta_{ik}, \phi_{ik}\right) = \exp\left(\frac{y_{ik}\theta_{ik} - A\left(\theta_{ik}\right)}{\phi} + C\left(y_{ik}, \phi\right)\right). \tag{9}$$

Then, the log-likelihood of the parameter θ_{ik} and ϕ is expressed as

$$l\left(\theta_{ik}, \phi; \mathbf{y}\right) = \log p\left(\mathbf{y} \mid \theta_{ik}, \phi\right) = \sum_{i,k} \frac{y_{ik}\theta_{ik} - A\left(\theta_{ik}\right)}{\phi} + C\left(y_{ik}, \phi\right). \tag{10}$$

The θ_{ik} is the canonical parameter and depends via a linear predictor on explanatory variables. The dispersion parameter is usually unknown and used to model the variance of the response data. The functions $A(.)$ and $C(.)$ are known and determined by the specified distribution of the family. Then, for Bernoulli distributed RR observations the natural form of the parameters is given by

$$\begin{aligned}
\theta_{ik} &= \log\left(\frac{c_{ik}+d_{ik}\tilde{\pi}_{ik}}{1-(c_{ik}+d_{ik}\tilde{\pi}_{ik})}\right) && (11)\\
A(\theta_{ik}) &= \log\left(1+\exp\left(\theta_{ik}\right)\right) && (12)\\
\phi &= 1. && (13)
\end{aligned}$$

Thus, the general properties of the exponential family distributions can be used to make inferences about the parameters. For instance, the maximum likelihood estimate for the population prevalence $\tilde{\pi}$ can be derived. The log-likelihood has a unique maximum at $\hat{\theta}$ which is the solution to

$$\begin{aligned}
\sum_{i,k} y_{ik} &= A'(\hat{\theta}) = \sum_{i,k}\left(1+\exp(-\hat{\theta}_{ik})\right)^{-1}\\
\sum_{i,k} y_{ik} &= \sum_{i,k}(c_{ik}+d_{ik}\hat{\pi})\\
\hat{\pi} &= \left(\sum_{i,k} y_{ik}-\sum_{i,k} c_{ik}\right)/\sum_{i,k} d_{ik} = (\overline{y}-\overline{c})/\overline{d}. && (14)
\end{aligned}$$

This maximum likelihood (ML) estimate for the prevalence is referred to as a weighted (ML) estimate, where the weights are defined by the RR parameters.

Furthermore, the inverse of the function $A'(\theta)$ represents the natural link function. Thus, the modified natural link function for Bernoulli distributed RR data can be derived using this property of the exponential family distribution:

$$\begin{aligned}
\pi_{ik} &= A'\left(\theta_{ik}\right)\\
&= \left[1+\exp\left(-\log\left(\frac{c_{ik}+d_{ik}g^{-1}(\eta_{ik})}{1-(c_{ik}+d_{ik}g^{-1}(\eta_{ik}))}\right)\right)\right]^{-1}\\
&= c_{ik}+d_{ik}g^{-1}(\eta_{ik})\\
\eta_{ik} &= g\left((\pi_{ik}-c_{ik})/d_{ik}\right),
\end{aligned}$$

where $g^{-1}(.)$ is the cumulative distribution function for (non-randomized) Bernoulli distributed data as defined in Equation (6). The corresponding link functions are defined in Equation (8).

ML and REML estimation

It can be shown that the maximum likelihood (ML) equations for the GLM for RR data resemble the general form of the GLM ML-equations (Fox et al., 2018, Appendix B). The only difference is that the conditional expected RR observation includes the RR parameters, and a modified link function is needed to link the conditional expected response to the linear term. The GLM parameters are usually estimated by ML methods using the iterative weighted least squares (IWLS) algorithm or Fisher scoring algorithm. The `glm` function implemented in R is a very flexible implementation of the general GLM framework (Chambers and Hastie, 1992). The package **GLMMRR** provides an expanded version of this function (`RRglm`), which includes modified link functions to fit GLMs on RR data. Given ML estimates, (maximum) likelihood theory can be used to obtain likelihood ratio tests, Wald and score tests.

The likelihood equations for the fixed effect parameters of the GLMM for RR data have the same structure as those for the GLMM (Fox et al., 2018, Appendix B). For the fixed effects, the GLM methodology can be used, since the fixed effects are not included in the random effect distribution. Depending on the dimension of the random effect parameter, numerical approximations are required to approximate the integrals to estimate the variance components and random effects. The numerical methods available in **GLMMRR** build on those available in the package **lme4**. Different numerical methods have been proposed; Laplace approximation and adaptive Gaussian quadrature are both implemented. Laplace approximation is usually fast and the default. The approximation improves when the cluster sizes increase. In Gaussian quadrature, a number of quadrature points need to be chosen, and the approximation is improved by increasing the number of quadrature points. Adaptive Gaussian quadrature usually fails when the dimension of the random effects is larger than two. It is also possible to compute restricted ML estimates (REML), which is also implemented. The `control` argument can be used in the function call to `RRglmer` to set the control parameters, which includes the optimizer to be used.

Randomized item-response theory modeling

An important class of item response theory (IRT) models belong to the GLMM class (Rijmen et al., 2003). These IRT models have a linear component for the transformed expected values of a binary response variable, which contains a random component(s) representing the latent variable(s). In the most common form, the linear term has a random component representing a random person effect and fixed components representing item effects. In Section 2.2.1, it is shown that the GLMM is extended to RR data using modified link functions. Therefore, IRT models belonging to the GLMM class can also be generalized to randomized item-response theory (RIRT) models.

The randomized item-response observations are clustered by persons and items, with c_{ik} and d_{ik} representing the RR parameters. The latent variable for person i is denoted by ϑ_i and the effect of item k by β_k. Then, the RIRT model – the Rasch model for RR data – can be represented as

$$\begin{aligned} \pi_{ik} &= c_{ik} + d_{ik} g^{-1}\left(\vartheta_i + \beta_k\right) \\ \vartheta_i &\sim N(0, \sigma^2), \end{aligned} \tag{15}$$

where g^{-1} is usually the logistic or cumulative normal distribution function. The item effects have a positive sign and should be interpreted as a easiness parameters.

The RIRT model can be fitted in R using the function `RRglmer` from our package **GLMMRR**. This is similar to fitting an IRT model using the function `glmer` from **lme4** (De Boeck et al., 2011), except that a modified link function is required for the RIRT. The data needs to be in a long format. Then, each data case has a response (`response`), a person identifier (`person`), an item identifier (`item`), the type of RR design (`RRmodel`), and RR design parameters (`RRp1,RRp2`). The RR model and parameters are allowed to vary across data cases.

To fit the RIRT model in Equation (15), a linear component is defined from the factor variables: `-1+item+(1|person)`. The `-1` restricts the general mean to zero, the `item` represents the item parameters, and the `(1|person)` represents the latent variable (random effect). This linear component is set equal to the outcome `response` in the model formula. To fit a logistic RIRT, the call to `RRglmer` includes a modified logistic link function (`RRlink.logit`):

```
RRglmer(response ~ -1 + item + (1|person), link = "RRlink.logit",
    RRmodel = RRmodel, p1=RRp1, p2=RRp2, data=data)
```

where the `RRmodel`, `p1`, and `p2` arguments define the RR design for every single response. The probit RIRT model can be fitted using the link function `RRlink.probit`. The RIRT models can be extended by including (1) item-covariate models (e.g., linear-logistic test model), (2) person-covariate models (e.g., multilevel IRT), and (3) person-by-item covariate models. Covariates with fixed or random effects can be included in the linear term.

Model fit and diagnostics

The likelihood ratio (LR) test can be used to compare nested models. The nested model is a restriction of a more general model by restricting one or more parameters most often to zero. The log-likelihood ratio (multiplied by minus two) is asymptotically chi-square distributed with the degrees of freedom equal to the difference in the number of free parameters. The LR test can be performed with the `anova` function to compare two nested GLM(M)s for RR data. Note that the LR test cannot be applied to test a hypothesis on the boundary of the parameter space. Since then the LR statistic is no longer chi-square distributed. ML estimation is preferred when comparing models that only differ in their fixed part.

To compare (non)-nested models, the usual information criteria can be used. The AIC and the BIC are both computed for GLM and GLMM, with the function `RRglm` and `RRglmer`, respectively, and reported in their output. The `anova` function also reports the AIC and BIC.

To evaluate the significance of fixed effects, a z-statistic is reported, which is the ratio of the parameter estimate and the estimated standard error. The z-statistic is asymptotically equivalent to the LR test. P-values are reported in the output of `RRglm` and `RRglmer` with the assumption that the z-statistic is asymptotically normally distributed.

Residuals

The error term in GLM(M)s represents a Bernoulli random error term, and the errors are assumed to be independently distributed. There are different types of residuals and different types of residual sum of squares to examine the fit of the model. The estimated residuals in the GLM and GLMM are based on the fitted RR probabilities, $\hat{\pi}_{ik}$. The fitted prevalence can be computed as $\hat{\tilde{\pi}}_{ik} = (\hat{\pi}_{ik} - c_{ik})/d_{ik}$, according to Equation (1).

GLM The residuals are computed for each observation y_{ik}, but for notational convenience, the index k is dropped, and the index i refers to a single observation y_i. The Pearson and deviance residual is often computed, and both are used in a goodness-of-fit statistic. The Pearson residual is defined as the standardized difference of the response and its expected value,

$$r_P(y_i, \hat{\pi}_i) = \frac{y_i - \hat{\pi}_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)}}.$$

The covariate patterns can be used to group the residuals in a natural way. Each unique covariate pattern defines a cluster j, where the number of response observations in cluster j is given by n_j, the observed proportion $\overline{y}$, and a unique estimated probability denoted as $\hat{\pi}_j$. Then, the Pearson residual for cluster j is defined as

$$r_P(\overline{y}_j, \hat{\pi}_j) = \frac{\overline{y}_j - \hat{\pi}_j}{\sqrt{\hat{\pi}_j(1-\hat{\pi}_j)/n_j}}.$$

The clustering of observations can also be based on the predicted outcome (Hosmer and Lemeshow, 1980). The observations are sorted according to their fitted probabilities, and this sorted vector is divided into J clusters of equal size. The corresponding clustered Pearson residuals are referred to as H-L residuals. The Pearson residuals and clustered Pearson and H-L residuals can be extracted from a fitted object of class `RRglm` with the following commands, respectively:

```
residuals(object, type="pearson")
residuals(object, type="pearson.grouped")
residuals(object, type="hosmer-lemeshow", ngroups=10)
```

For the H-L residuals, the `ngroups` argument defines the desired number of clusters, and the default is 10. The sum of squared grouped residuals defines a goodness-of-fit statistic, which is given by

$$X^2 = \sum_{j=1}^{J} r^2(\overline{y}_j, \hat{\pi}_j) = \sum_{j=1}^{J} \frac{n_j\left(\overline{y}_j - \hat{\pi}_j\right)^2}{\hat{\pi}_j(1-\hat{\pi}_j)}.$$

Depending on the type of clustering, the statistic is the Pearson goodness-of-fit statistic, X_p^2, or the H-L goodness-of-fit statistic, X_{HL}^2. The Pearson statistic is asymptotically chi-square distributed for a fixed number of groups with degrees of freedom $J-(q+1)$, and q the number of independent covariates. When the predictor variables are continuous, the X_p^2 test cannot be used. For semi-continuous observations, it is possible to find a clustering with a sufficient number of observations in each cluster. The X_{HL}^2 statistic is assumed to be chi-square distributed with $J-2$ degrees of freedom (Hosmer and Lemeshow, 1980). The X_p^2 and X_{HL}^2 statistics can be computed from a fitted object of class `RRglm` using the function `RRglmGOF`:

```
RRglmGOF(object, doPearson = TRUE, doHlemeshow = TRUE,
    hlemeshowGroups = 10, rm.na = TRUE)
```

The default number of clusters for the X_{HL}^2 statistic is ten, and data cases with missing observations are excluded. The deviance residual is defined as

$$r_D(\bar{y}_i, \hat{\pi}_i) = sign(\bar{y}_i - \hat{\pi}_i)\sqrt{2n_i\left(\bar{y}_i log\left(\frac{\bar{y}_i}{\hat{\pi}_i}\right) + (1-\bar{y}_i) log\left(\frac{1-\bar{y}_i}{1-\hat{\pi}_i}\right)\right)}, \tag{16}$$

where $sign(\bar{y}_i - \hat{\pi}_i) = 1$ when $\bar{y}_i \geq \hat{\pi}_i$ and -1 when $\bar{y}_i < \hat{\pi}_i$. The grouped deviance residual is defined for cluster i with $n_i > 1$ and denoted as $r_d(\overline{y}_i, \hat{\pi}_i)$. For clustered observations, the sum of squared deviance residuals is considered to be a goodness-of-fit statistic, denoted as X_d^2, which is asymptotically chi-square distributed with $J-(q+1)$ degrees of freedom. The deviance residuals, the grouped deviance residuals, and the deviance goodness-of-fit statistic can be extracted from a fitted `RRglm` object with the commands, respectively:

```
residuals(out, type = "deviance")
residuals(out, type = "deviance.grouped")
RRglmGOF(object, doDeviance = TRUE, rm.na = TRUE)
```

In the output of `RRglmGOF`, the goodness-of-fit test(s) are reported, the p-value(s), the degrees of freedom, and the number of groups. For the H-L test, the results are also given for each cluster. The deviance statistic is equivalent to the LR statistic for testing the fitted model against the saturated model, which has a deviance of zero. The Pearson goodness-of-fit statistic is a score test statistic, also testing the fitted model against the saturated model. Thus, the Pearson and the deviance statistic are the score test and LR test for GLMs, respectively.

GLMM Different residuals can be considered for GLMMs. The response residual conditional on the random effect is defined as the difference between the observation and the conditional expected value:

$$\begin{aligned} r_c(y_{ik}, \hat{\pi}_{ik}) &= y_{ik} - E(y_{ik} \mid \mathbf{x}_{ik}, \mathbf{z}_{ik}, \mathbf{b}_i) \\ &= y_{ik} - \pi\left(\mathbf{x}_{ik}^t \hat{\boldsymbol{\beta}} + \mathbf{z}_{ik}^t \hat{\mathbf{b}}_i\right). \end{aligned} \tag{17}$$

It can be extracted from the `residual` function by providing the argument `type="response"`. The unconditional response can be computed by integrating out the random effects to obtain the difference between the observation and the marginal mean. This residual is computed with the argument `type="unconditional.response"`.

The conditional and unconditional response residuals can be standardized by dividing them by their standard deviation, which leads to Pearson residuals:

```
residuals(object, type = "pearson")
residuals(object, type = "unconditional.pearson")
```

Finally, as a result of sustained compatibility with **lme4**, the usage of residuals aimed at `"merMod"` objects, such as working residuals and conditional deviance residuals, is not limited. Pearson residuals can also be scaled by any given user-specified weights.

Software

The package **GLMMRR** contains two main functions, `RRglm` and `RRglmer`, for fitting a GLM and a GLMM given RR data, respectively. Both functions include the four link functions (logit, probit, cloglog, cauchit) for the different RR designs. The function `RRglm` makes a call to the function `glm` with the appropriate link function to fit a GLM for (binary) RR data. In the same way, the function `RRglmer` makes a call to the function `glmer` to fit a GLMM for (binary) RR data. The fit of both models, GLM and GLMM, is arranged by the computational routines of `glm` and `glmer`. Their general control parameters, the model and data-checking options, the type of optimizer, number of iterations can be specified in the `RRglm` and `RRglmer` functions. Thus, the numerical optimization algorithm and its specification can be defined in a similar way as in the call to functions `glm` and `glmer`.

The function `RRglm` and `RRglmer` creates an object of class `"RRglm"` and `"RRglmer"`, respectively. The package's summary, print and plot function, can be used to get estimation results from an object of each class. Data needs to be defined in long format, where an RR specification is needed for every single case, including the type of RR model and the design parameters. The package allows for different RR models and different design parameters across data cases. Together with a binary RR (outcome) variable and possible predictors, a GLM can be fitted. When also including a (factor) cluster variable, implying a correlation among clustered observations, a GLMM can be fitted. The complete functionality of the package can be accessed by making further input specifications.

Input

- `RRglm`
  The general call to the function is: `RRglm(formula,link,item,RRmodel,p1,p2,data,na.action = "na.omit",...)`
  - formula: a two-sided linear formula object describing the model to be fitted, with the response on the left of a ∼ operator and the terms, separated by + operators, on the right.
  - link: a GLM RRlink function for binary outcomes. Must be a function name; `RRlink.logit`, `RRlink.probit`, `RRlink.cloglog`, and `RRlink.cauchit`.
  - item: optional item identifier to obtain prevalence estimates per level of item.
  - RRmodel: the RR model per data case. Available options: `DQ`, `Warner`, `Forced`, `UQM`, `Crosswise`, `Triangular`, and `Kuk`.

- p1: the RR parameter p_1, defined per data case ($0 \leq p_1 \leq 1$).
- p2: the RR parameter p_2, defined per data case ($0 \leq p_2 \leq 1$).
- data: a data frame containing the variables named in formula as well as the RR model and parameters. If the required information cannot be found in the data frame, or if no data frame is given, then the variables are taken from the environment from which RRglm is called.
- na.action: a function that indicates what should happen when the data contain NAs. The default action (na.omit, as given by getOption("na.action"))) strips any observations with any missing values in any variables.

• `RRglmer`
  The general call to the function is `RRglmer(formula,item,link,RRmodel,p1,p2,data,control = glmerControl(),na.action = "na.omit",...)`

  - formula: a two-sided linear formula object describing both the fixed and random effects part of the model, with the response on the left of a ∼ operator and the terms, separated by + operators, on the right. Random-effect terms are distinguished by vertical bars ("|") separating expressions for design matrices from grouping factors.
  - item : optional item identifier to obtain prevalence estimates per level of item.
  - link: a GLM RRlink function for binary outcomes. Must be a function name; `RRlink.logit`, `RRlink.probit`, `RRlink.cloglog`, and `RRlink.cauchit`.
  - RRmodel: the RR model per data case. Available options: `DQ`, `Warner`, `Forced`, `UQM`, `Crosswise`, `Triangular`, and `Kuk`.
  - p1: the RR parameter p_1, defined per data case ($0 \leq p_1 \leq 1$).
  - p2: the RR parameter p_2, defined per data case ($0 \leq p_2 \leq 1$).
  - data: a data frame containing the variables named in formula as well as the RR model and parameters. If the required information cannot be found in the data frame, or if no data frame is given, then the variables are taken from the environment from which RRglmer is called.
  - na.action: a function that indicates what should happen when the data contain NAs. The default action (na.omit, as given by getOption("na.action"))) strips any observations with any missing values in any variables.
  - control: a list (of correct class, resulting from `lmerControl()` or `glmerControl()`, respectively) containing control parameters, including the nonlinear optimizer to be used and parameters to be passed through to the nonlinear optimizer; see the `lmerControl documentation` for details.

Output

• `RRglm` An object of class `RRglm`, which extends the class `glm` with RR data. The object of class `RRglm` contains the regular GLM output and the following components:

  - Item: the item levels for each data case – prevalence rates are computed per level of item.
  - RRc: the RR-parameter c for each data case (Equation (1)).
  - RRd: the RR-parameter d for each data case (Equation (1)).
  - RRmodel: the RR model for each data case.
  - RRp1: the RR design parameter p_1.
  - RRp2: the RR design parameter p_2.

• `RRglmer` An object of class `"RRglmerMod"`, which extends the class `"glmerMod"` with RR data. Many methods are available for the general class `"merMod"` to which `"glmerMod"` and also the class `"RRglmerMod"` belongs. The methods for the `"merMod"`-class can be found in the documentation of **lme4** and they are applicable to an object from class *"RRglmerMod"*.

Applications

An RR validation study

Höglinger and Jann (2018) conducted an online experiment to validate different RRTs on the platform Amazon Mechanical Turk (https://boris.unibe.ch/81516). Participants were asked to play one of the two dice games, after which they were asked if they played honestly using randomly one of four RRTs. In the roll-a-six game, the participant rolled a digital die by clicking a button and was asked if the first roll resulted in a six. In the prediction game, the participant was asked to think of a number, roll the digital die, and was asked if the outcome corresponded to the memorized prediction. The participants were asked if they won in the dice game, and since both games relied on self-reports about the dice-roll outcomes, cheating was easily possible. However, for the roll-a-six game, the virtual dice outcomes were registered, and it was evaluated if participants illegitimately claimed a $2 bonus payment. This enabled classifying the participants as `cheater` or `honest` (non-cheater) players. In the prediction game, individual cheating was not detectable. However, it was expected that around one-sixth of all predictions were correct since the dice outcomes were random. A systematic deviation from this percentage was attributed to cheating.

To disguise the true purpose of the study, the survey was posted as a survey on "Mood and Personality" and included a range of questions, for instance, questions on the big five personality trades. Participants were randomly appointed to one of four questioning methods (DQ, CW, FR, UQ) and were asked four sensitive questions. Besides the question about honest playing in the dice game, questions were asked about shoplifting, tax evasion, and voting. Although the answers to these last three questions could not be validated, the prevalence estimates of those who were identified as cheaters and non-cheaters in the dice games can be compared across RRTs.

Höglinger and Jann (2018) compared prevalence estimates of cheating in both games to those computed from the answers to the question of whether they played honestly using the four RR techniques. They concluded that two RR techniques (FR, UQ) performed similarly to DQ and did not reduce the level of misreporting. The level of underreporting was reduced by CW. However, CW also increased the level of overreporting, and the corresponding prevalence estimates were substantially higher than the true prevalence estimates of cheating.

The **GLMMRR** is used to do a joint regression analysis of the RR data of the four questioning techniques. In a joint analysis, the levels of under- and overreporting across RRTs can be directly compared through an interaction analysis of identified cheaters and the RRTs. The joint analysis is needed to quantify the different levels of underreporting across RRTs, and to identify who was misreporting. It is also examined if any background variables explain differences in misreporting across RRTs and sensitive questions.

The experiment had a two-by-three-by-five factorial design. Factor one represented the type of dice game (prediction game, roll-a-six game). Factor two was the type of RRT to ask the sensitive questions (DQ, CW, FR, UQ). The third factor was included to examine whether the implementations of the random devices produced the expected outcome distributions. This third factor is integrated into the current study by defining RRTs with different RR design parameters. Höglinger and Jann (2018) discuss the RRTs (factor 2), the random devices (factor 3), and the corresponding RR design parameters. Their supplement *Documentation and codebook of the survey* was used to prepare the raw data for this study for analysis in R, which includes the specification of the design parameters for the RRTs. Our supplementary R script 'ASQ-MTurk data.R' comprises the code for preparing the data MTURK. The package **GLMMRR** contains the prepared data object MTURK, which is constructed from the raw data using the code of the supplementary R script.

The true prevalence estimates of cheating are estimated for the roll-a-six game (`dicegame=2`) for the different RRTs. The estimates are based on the discrepancy between the actual dice outcome and the participant's response to whether the first roll was a six, which was asked through direct questioning. The true prevalence estimates are reported for each of the groups assigned to one of the RRTs.

```
R> package(GLMMRR)
R> data("MTURK", package="GLMMRR")
R> by(MTURK$cheaterdc[MTURK$dicegame==2],
+    MTURK$RRmodel[MTURK$dicegame==2],mean,na.rm=TRUE)

MTURK$RRmodel[MTURK$dicegame == 2]: DQ
[1] 0.04450262
--------------------------------------------------------------------------------
MTURK$RRmodel[MTURK$dicegame == 2]: Crosswise
[1] 0.06032787
```

```
-----------------------------------------------------------------------------
MTURK$RRmodel[MTURK$dicegame == 2]: UQM
[1] 0.05004812
-----------------------------------------------------------------------------
MTURK$RRmodel[MTURK$dicegame == 2]: Forced
[1] 0.05188067
```

Around 5% of the participants cheated in the roll-a-six game, and the estimates are comparable across RRT groups. It is examined if the prevalence estimates can also be recovered from the participant's responses to the question if they honestly reported whether a six was rolled (item `honest dice game reporting`, where the response is coded as honest=0 and dishonest=1). This question was asked to all participants, but different RRTs were used to obtain the response. The object is to validate the RRTs by comparing the prevalence estimate of cheating with the true prevalence estimate. The prevalence estimates are computed using the `RRglm` function where the `item` is the sensitive question if they are reported honestly. The design parameters are stored in the data variables `RRp1` and `RRp2`, which are also reported in the output. It can be seen that, for instance, the CW design has two sets of parameters, where some participants were questioned with the CW method and design parameters .16 and 0, and others with .20 and 0.

```
R> MTURK_cheating1 <- MTURK[which(MTURK$dicegame==2 &
+   MTURK$Question=="cheating dice game"),]
R> rolla6 <- RRglm(RR_response ~ 1, item = Question,
+   link = "RRlink.logit", RRmodel = RRmodel,
+   p1=RRp1,p2=RRp2,data = MTURK_cheating1)
R> summary(rolla6)

### GLMMRR - Binary Randomized Response Data ###
Generalized linear fixed-effects model

Family:                         binomial
Link function:                 RRlogit

---------------------------------------------------------
Item:                          cheating dice game
Model(s):                  DQ (1.00 | 0.00)
                          Crosswise (0.16 | 0.00) (0.20 | 0.00)
                          UQM (0.78 | 0.49) (0.78 | 0.52)
                          Forced (0.75 | 0.67)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.143393   0.0204156 1142
        DQ          0.039370   0.0099632  381
    Forced         -0.019361   0.0172690  769
       UQM          0.053496   0.0165870  778
```

For the question *honest dice game reporting*, the estimated population proportion of dishonest reporters (ML estimate with standard errors) is reported for each RR design. Each RRT estimate is the (weighted) average across different design parameters. It follows that the prevalence estimate of the CW method overestimates the true value of 6% of detected cheaters who were CW questioned. Furthermore, the DQ and UQ perform approximately the same. The estimate is even negative for the FR technique. This can occur when some participants did not follow the RR instructions and/or when the random device distribution deviates from the expected distribution. This could also be an underlying problem of the CW method. A total of 33.6% admitted that they do not know exactly the birthday of their parents. This could bias their response to the unrelated question, which stated if their father/mother's birthday was in January or February (with an expected probability of 15.9%), or between the 1st and the 6th of the month (with an expected probability of 19.7%). The overestimation of the CW method could be caused by honest reporters who incorrectly answered `No` to the unrelated question about the birthday of one of their parents.

Misreporting is investigated further by computing the reduction in misreporting for each RRT for detected cheaters (dishonest reporters) and non-cheaters (honest reporters). Therefore, a dummy coded variable is defined for the cheaters (`cheaterdc=1`) and for each of the RRTs (DQ, CW, UQ, FR). Logistic regression analysis is performed using the `RRglm` function for the RR data of item `honest dice game reporting` (variable `RR_response`) in the roll-a-six game conditional on the RRT and cheater identifiers.

```
R> rolla6A <- RRglm(RR_response ~ 1 + cheaterdc + CW + UQ + FR +
+   cheaterdc*CW+cheaterdc*UQ, item = Question,
+   link = "RRlink.logit", RRmodel = RRmodel, p1=RRp1, p2=RRp2,
+   data = MTURK_cheating1, na.action = "na.omit")
R> summary(rolla6A)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -4.8807     0.6043  -8.077 6.63e-16 ***
cheaterdc      5.8302     0.8087   7.209 5.63e-13 ***
CW             2.8283     0.6375   4.437 9.14e-06 ***
UQ             1.3453     0.8399   1.602   0.1092
FR            -1.3582     0.6941  -1.957   0.0504 .
cheaterdc:CW  -3.6246     0.9143  -3.964 7.36e-05 ***
cheaterdc:UQ  -2.0967     1.0798  -1.942   0.0522 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3988.2  on 3069  degrees of freedom
Residual deviance: 2631.5  on 3063  degrees of freedom
  (5 observations deleted due to missingness)
AIC: 2645.5

Number of Fisher Scoring iterations: 7
```

The reference level is the DQ technique (factor `RRmodel`) for those who were identified as honest reporters (factor `cheaterdc`) responding to being dishonest in the roll-a-six game. For DQ, the odds for dishonest reporting for non-cheaters is around zero ($\exp(-4.88) = 0.008$), and for cheaters around 2.58. Of the DQ participants, identified as cheaters, the probability of admitting to being dishonest about whether a six was rolled is around 72%. Participants might have guessed that this type of misreporting could be easily detected, which led to a non-substantial level of misreporting under DQ. The odds for dishonest reporting for non-cheaters in the UQ and FR condition are also close to zero. The UQ and FR effects are also not significantly different from zero. For the non-cheaters, the response technique, DQ, UQ, or FR, did not influence their response. However, in the CW condition, the odds of dishonest reporting for non-cheaters is significant and around .13, with a probability of 11.3% of dishonest reporting while being identified to be honest reporters. This led to the overestimation of the true prevalence rate by CW.

The opposite occurred for the cheaters in the CW and UQ condition. It was expected that the cheaters would be honest about their dishonesty in the roll-a-six game without the risk of disclosure in the CW and UQ condition. Instead, for both techniques, the admitted level of misreporting by cheaters decreased. For cheaters in the CW and UQ condition, the probability of being dishonest is around 53.8% and 54.9%, respectively, which is much less than the 72% under DQ. Höglinger and Jann (2018) argued that some cheaters might have misused the RRT to answer untruthfully without risk of detection who would have felt compelled to answer truthfully in DQ. It turned out that it was not possible to estimate an interaction effect for cheaters in the FR condition.

The three other sensitive questions (voting, shoplifting, tax evasion) do not have the problem that participants might be suspicious of being disclosed for their dishonesty as in the roll-a-six game. However, it is expected that the prevalence rates will be lower for the DQ group. Their responses are not masked, and those participants are tended to underreport the questioned behavior in comparison to those not under risk of disclosure through the RR questioning techniques. Differences in prevalence rates across sensitive items (factor `Question`) and question techniques (factor `RRmodel`) are explored through a logistic regression using the `RRglm` function. (factor `Question`)

```
R> MTURK_cheating2 <- MTURK[which(MTURK$dicegame==2 &
+   MTURK$Question!="cheating dice game"),]
R> rolla6B <- RRglm(RR_response ~ 1 + RRmodel + Question +
+   cheaterdc*RRmodel, item = Question, link = "RRlink.logit",
+   RRmodel = RRmodel,p1=RRp1,p2=RRp2,data = MTURK_cheating2)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               -0.87793    0.08118 -10.815  < 2e-16 ***
RRmodelCrosswise           0.26976    0.09415   2.865  0.00417 **
```

```
RRmodelUQM                 0.39283    0.09279   4.233  2.3e-05 ***
RRmodelForced              0.11554    0.09783   1.181  0.23762    
Questionshoplifting        0.61299    0.07040   8.708  < 2e-16 ***
Questiontax evasion       -1.01421    0.09100 -11.145  < 2e-16 ***
cheaterdc                  0.23951    0.31801   0.753  0.45135    
RRmodelCrosswise:cheaterdc -0.04895   0.40461  -0.121  0.90371    
RRmodelUQM:cheaterdc      -0.30170    0.42096  -0.717  0.47355    
RRmodelForced:cheaterdc    0.32900    0.42022   0.783  0.43367    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12705  on 9207  degrees of freedom
Residual deviance: 11797  on 9198  degrees of freedom
  (21 observations deleted due to missingness)
AIC: 11817

Number of Fisher Scoring iterations: 4
```

The intercept corresponds to non-cheaters in the DQ condition responding to the item non-voting. For DQ, the odds for non-voting of the non-cheaters is around .41 ($\exp(-0.88) = 0.41$), and for cheaters, around 0.53. However, the cheaters did not respond significantly differently from the non-cheaters as they did in reporting about being dishonest in the dice game. The prevalence rates of those questioned with privacy protection (CW, UQ, FR) are higher – the CW and UQ rates are significantly higher – than those questioned directly. The corresponding odds ratios are .54, 62, and .47 for CW, UQ, and FR, respectively. The prevalence rates for shoplifting are significantly higher and for tax evasion significantly lower than for non-voting. It is apparent that cheaters do not report significantly differently under a privacy-protected response technique in comparison to the non-cheaters since the interaction effects are approximately zero and non-significant. This occurred when cheaters were asked about their dishonesty in the roll-a-six game. This was probably provoked by the knowledge that their dishonesty could be detected under DQ. For the questions about non-voting, shoplifting, and tax evasion, it was known that this was not possible, so cheaters were not/less inclined to misreport in the non-DQ condition.

The weighted prevalence rate (averaged over results from the same questioning technique with different design parameters) for each item and questioning technique are reported in the output. It follows that the prevalence rates are lower for DQ than for the other RRTs, but the differences are small. Note that the reported significance in rates between CW and DQ and between UQ and DQ were averaged across the three items.

```
### GLMMRR - Binary Randomized Response Data ###
Generalized linear fixed-effects model

Family:                          binomial
Link function:                  RRlogit

---------------------------------------------------------
Item:                          non voting
Model(s):                 DQ (1.00 | 0.00)
                         Crosswise (0.16 | 0.00) (0.20 | 0.00)
                         UQM (0.78 | 0.49) (0.78 | 0.52)
                         Forced (0.75 | 0.67)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise           0.38215    0.022718 1142
        DQ           0.30607    0.023673  379
    Forced           0.33333    0.023720  768
       UQM           0.33889    0.022338  776

---------------------------------------------------------
Item:                          shoplifting
Model(s):                 DQ (1.00 | 0.00)
                         Crosswise (0.16 | 0.00) (0.20 | 0.00)
                         UQM (0.78 | 0.49) (0.78 | 0.52)
                         Forced (0.75 | 0.67)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise           0.46339    0.022914 1145
        DQ           0.44357    0.025452  381
    Forced           0.47479    0.024016  769
       UQM           0.56174    0.022936  778

---------------------------------------------------------
Item:                             tax evasion
Model(s):                 DQ (1.00 | 0.00)
                         Crosswise (0.16 | 0.00) (0.20 | 0.00)
                         UQM (0.78 | 0.49) (0.78 | 0.52)
                         Forced (0.75 | 0.67)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise           0.18850    0.021098 1143
        DQ           0.11549    0.016374  381
    Forced           0.13057    0.021182  771
       UQM           0.19407    0.020301  775
```

The inclusion of the interaction effect between cheating status and RRT can be tested through a model comparison. The model is fitted without the interaction term (object `rolla6C`), and the `anova` function is used to produce a deviance table for the fitted objects. It follows that the interaction term does not lead to a significant model improvement.

```
R> anova(rolla6C, rolla6B,test="Chisq")
Analysis of Deviance Table

Model 1: RR_response ~ 1 + RRmodel + Question
Model 2: RR_response ~ 1 + RRmodel + Question + cheaterdc * RRmodel
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      9202      11802
2      9198      11797  4   5.3509   0.2532
```

Residual analysis

The fit of the GLM with factor variables `RRT` and `Question` (object `rolla6C`) is evaluated with a residual analysis. The function `RRglmGOF` is used to compute overall goodness-of-fit tests. The Pearson and deviance chi-square statistics are computed given a grouping of the observations based on the predictor variables. The H-L statistic is computed for ten groups of (approximately) equal size. There is evidence for a lack-of-fit when the statistic values are large.

```
R> RRglmGOF(RRglmOutput = rolla6C, doPearson = TRUE, doDeviance = TRUE,
+   doHlemeshow = TRUE)

GLMMRR - Binary Randomized Response Data

Goodness-of-Fit Testing
Response variable:                RR_response
Predictor(s):                        RRmodel Question
Entries dataset:                 9208
---------------------------------------------------------
Summary:

                Statistic P.value df Groups
Pearson         12.95     0.0438  6  12
Deviance        13.01     0.0429  6  12
Hosmer-Lemeshow  8.82     0.3578  8  10

---------------------------------------------------------
```

It follows that the Pearson and deviance goodness-of-fit statistics show a lack of fit, whereas the H-L statistic does not show a lack of fit. To further investigate the fit, the H-L statistic is computed for the same model but with different link functions. A different link function can improve the fit of the model and reduce the effects of a misspecified linear predictor. The estimated statistic values are given for the probit link function (see supplementary R-code for the complete analysis), which shows only a slight improvement in fit. The decrease in AIC is small and around .62.

```
R> rolla6E <- RRglm(RR_response ~ 1 + RRmodel + Question,
+   item = Question, link = "RRlink.probit",RRmodel = RRmodel,
+   p1=RRp1,p2=RRp2,data = MTURK_cheating2)
R> RRglmGOF(RRglmOutput = rolla6E, doPearson = TRUE, doDeviance = TRUE,
+   doHlemeshow = TRUE) ## improved fit over logit link

--------------------------------------------------------
Summary:

                Statistic P.value df Groups
Pearson         12.35     0.0545  6  12
Deviance        12.40     0.0537  6  12
Hosmer-Lemeshow  8.69     0.3689  8  10

--------------------------------------------------------
```

Finally, the fitted probabilities are plotted against the estimated Pearson residuals for each RRT (R-code below) – it is also possible to plot the object of class "RRglm", `plot(rolla6C,which = 3,type = "pearson")`. In Figure 1, the Pearson residuals (filled circles) under the CW method are large for the zero response observations. Typical for CW, high fitted probabilities (prevalence rates) correspond to zero and to one response. There are three items, which leads to three different fitted probabilities for DQ. CW was used with two different sets of design parameters, which led to more than three fitted probabilities. The differences between prevalence rates and Pearson residuals for different design parameters are much larger for CW than for UQ and FR. For UQ and FR, the estimated Pearson residuals and fitted probabilities hardly differ across design parameters. It follows that the CW method is particularly sensitive to deviations from the design parameters. For an observed prevalence rate above .50, increasing the design parameter for CW will decrease the prevalence estimate. A possibility is that the design parameter for CW was incorrect and too low, which led to the overestimation of the prevalence rate for CW in the roll-a-six game. The residuals are also relatively large for low fitted probabilities, which is typical for DQ when the true prevalence rate is small. For reasons of brevity, we have omitted a further improvement of the model and to explain individual differences in prevalence by including individual predictor variables (e.g., gender, education, Big Five personality traits).

```
R> set <- names(rolla6C$linear.predictors)
R> dataset <- MTURK_cheating2[set,]
R> plot(rolla6C$fitted.values,residuals(rolla6C, type = "pearson"),
+   cex=.8,bty="l",xlim=c(.1,.8),ylim=c(-3,3),xlab="Fitted",
+   ylab="Residual (Pearson)")
R> points(rolla6C$fitted.values[which(dataset$DQ==1)],
+   residuals(rolla6C, type = "pearson")[which(dataset$DQ==1)],
+   xlim=c(0,1),ylim=c(-1,1),pch=15,col="black")
R> points(rolla6C$fitted.values[which(dataset$CW==1)],
+   residuals(rolla6C, type = "pearson")[which(dataset$CW==1)],
+   pch=16,col="green")
R> points(rolla6C$fitted.values[which(dataset$UQ==1)],
+   residuals(rolla6C, type = "pearson")[which(dataset$UQ==1)],
+   pch=17,col="red")
R> points(rolla6C$fitted.values[which(dataset$FR==1)],
+   residuals(rolla6C, type = "pearson")[which(dataset$FR==1)],
+   pch=18,col="blue")
R> legend(.6,3,c("DQ","CW","FR","UQ"),col=c("black","green",
+   "blue","red"),pch = c(15,16,18,17), bg = "gray95",cex=.7)
```

Extended item response modeling: Prevalence of student misconduct

Höglinger et al. (2014) performed an online survey to estimate the prevalence of various forms of student misconduct (e.g., plagiarizing, cheating in exams). RRT was used since students might be reluctant to reveal this kind of information. Four different types of RRTs were used (DQ, FR, CW,

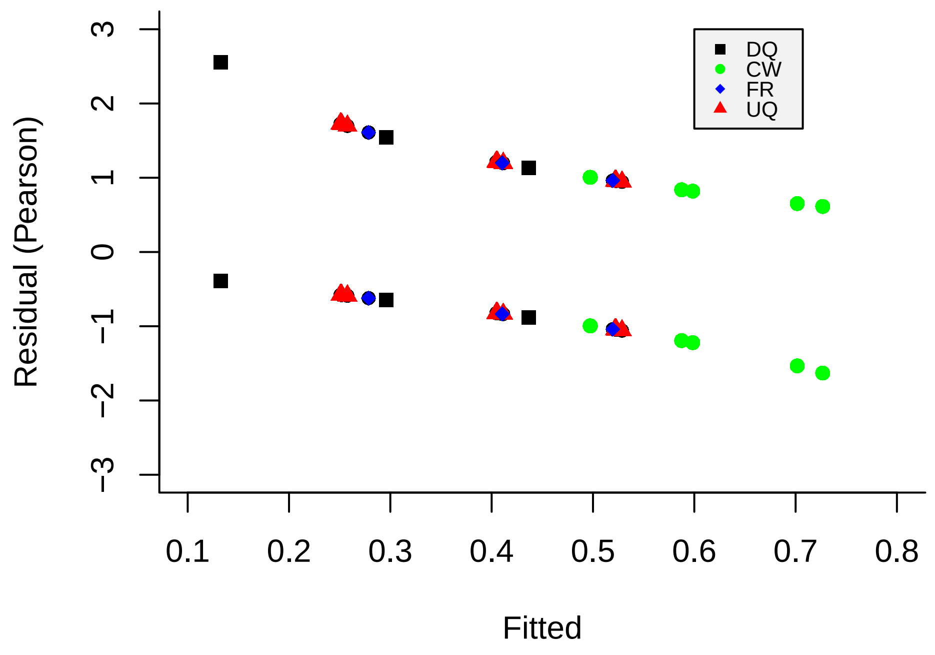


Figure 1: The roll-a-six game: for each RRT fitted success probabilities against Pearson residuals.

UQ). The RRTs were tailored to be implemented online which led to two implementations of FR (pick-a-numer, random wheel) and of CW (unrelated question, random wheel). In total, there were six different RRTs to which participants were randomly assigned: DQ, two implementations of FR, UQ, and two implementations of CW. For all RRT implementations, the level of protection was also varied, which led to different design probabilities. In total, six different sets of parameters for UQ, two for FR, and eight for CW. The package **GLMMRR** was used to do a joint regression analysis of the randomized item-response data, (1) to measure differences between RRTs, and (2) to examine differences in prevalence rates across conditions and students.

No.	Item
1	copied from other students during an exam (copied)
2	used illicit crib notes in an exam (crib notes)
3	used prescription drugs to enhance your performance (drugs)
4	handed in someone else's work without citing (plagiarism)
5	had someone else write a large part of a submitted paper (someone else's work)

Table 1: Five items for measuring prevalence of student misconduct.

Five sensitive items about student misconduct were surveyed using the six different RRTs with different design parameters. In Table 1, the five items are given about respondents' own misconduct during exams and submitting a paper. The items were assumed to measure student misconduct. An RIRT model is used to measure each student's level of misconduct given RR data while taking into account that students were assigned to different RRT conditions. A multiple-group (normal ogive) RIRT model is fitted, where the groups represent the RRT conditions. The function `RRglmer` of **GLMMRR** is used with factor variable `Question` and factor variable `expcond` representing the items and experimental conditions (RRTs), respectively, for the randomized responses (`RR_response`). The input `item` equals factor variable `Question` to obtain the (weighted) prevalence estimates for each item.

The latent variable is represented by the student identifier `id`, where each student responded to the five items. This latent variable represents a student's propensity to misconduct on an exam measured by the five items in Table 1. This (continuous) personality trait is measured with the

RIRT model with data collected under different RRT designs.

```
 R> out.re <- RRglmer(RR_response ~ 1 + Question + expcond + (1|id),
+   item=Question, link = "RRlink.probit", RRmodel = RRmodel,
+   p1=p1,p2=p2,data = ETHBE, control=glmerControl(
+   optimizer="bobyqa", optCtrl = list(maxfun = 200000)))
R> summary(out.re)

     AIC      BIC   logLik deviance df.resid
 21015.5  21103.2 -10496.7  20993.5    21394

Random effects:
 Groups Name        Variance Std.Dev.
 id     (Intercept) 0.4342   0.659
Number of obs: 21405, groups:  id, 4281

Fixed effects:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -1.20625    0.05633 -21.416  < 2e-16 ***
Questioncrib notes                    -0.34520    0.04582  -7.534 4.93e-14 ***
Questiondrugs                         -0.92016    0.05612 -16.395  < 2e-16 ***
Questionhanded in plagiarism          -0.83103    0.05369 -15.478  < 2e-16 ***
Questionhanded in someone else's work -1.06092    0.05982 -17.736  < 2e-16 ***
expcondFR pick-a-number                0.94880    0.06524  14.543  < 2e-16 ***
expcondCM pick-a-number                0.53036    0.08064   6.577 4.80e-11 ***
expcondFR random wheel                 0.99941    0.06520  15.329  < 2e-16 ***
expcondUQ Benford                      0.39748    0.07245   5.487 4.10e-08 ***
expcondCM unrelated question           0.67867    0.08017   8.465  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The levels of factor variable `Question` represent the item difficulties, where the intercept represents the first item (copied work). The levels of factor variable `expcond` represent the mean level of each RRT condition, and the intercept is the level of the DQ group. The latent variable variance in student misconduct is around 0.43, which shows that conditional on the item and experimental condition differences, around 30% ($= .43/(1 + .43)$) of the variance can be attributed to individual differences. The latent variable estimates represent the levels of student misconduct.

It is expected that students underreport their behavior. With an RRT individual responses are masked, and students are expected to answer more truthfully. Although students were randomly assigned to RRT conditions, the prevalence rates appear to be different across RRTs. The package **multcomp** is used to test individual null hypotheses representing differences between RRTs. Linear combinations of the experimental condition parameters are defined and tested simultaneously. Four hypotheses are evaluated to test the difference in prevalence rates (1) between the two FR implementations (pick a number, random wheel), (2) between the two CW implementations (pick-a-number, unrelated question), (3) between FR and CW, (4) between FR and UQ. A matrix K is defined which represents the contrasts of interest and the function `glht` is used to perform the general linear hypothesis testing.

```
R> K <- matrix(c(c(0, 0, 0, 0, 0, 1, 0, -1, 0 , 0),
+             c(0, 0, 0, 0, 0, 0, 1, 0, 0 ,-1),
+             c(0, 0, 0, 0, 0, 1, -1, 1, 0 ,-1),
+             c(0, 0, 0, 0, 0, 0, 0, 1, -1 ,0)), nrow=4,ncol=10,byrow=T)
R> t <- glht(out.re, linfct = K)
R> summary(t)

         Simultaneous Tests for General Linear Hypotheses

Linear Hypotheses:
       Estimate Std. Error z value Pr(>|z|)
1 == 0 -0.05061    0.06176  -0.819    0.870
2 == 0 -0.14831    0.09125  -1.625    0.343
3 == 0  0.73917    0.11108   6.654   <0.001 ***
4 == 0  0.60193    0.06963   8.645   <0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Adjusted p values reported -- single-step method)
```

It follows that the different implementations of FR do not lead to different prevalence rates. Both FR implementations provide a similar level of privacy protection. Furthermore, both CW implementations have similar prevalence rates. For FR and CW, differences between random devices do not lead to different prevalence estimates. The difference is significant between FR and CW, where FR produces higher rates than CW. FR also produced higher rates than UQ. It is remarkable that in this study, students reported much higher rates with FR than CW, whereas in the RR validation study, it was the other way around. It is not clear why with FR, higher rates were obtained in the student misconduct study than with CW in comparison to the roll-a-six validation study. However, in the roll-a-six game, the prevalence rates with DQ also differed not much with those from FR.

The prevalence rates for the five items are estimated for each RRT, averaging across results from different design parameters. In the output, an overview is given of the RRTs which were used to administer each item. For each RRT, the design parameters are given which were used. For instance, for item `copied`: DQ has one set of design parameters, UQ six sets of design parameters, FR two sets of design parameters, and CW eight sets of design parameters. Note that different design parameters were used for different implementations of a random online device (Höglinger et al., 2014).

```
### GLMMRR - Binary Randomized Response Data ###
Generalized linear mixed-effects model

Family:                          binomial
Link function:                  RRprobit

---------------------------------------------------------
Item:                           copied
Model(s):                  DQ (1.00 | 0.00)
                          UQM (0.70 | 0.49) (0.70 | 0.50) (0.70 | 0.52)
                              (0.78 | 0.49) (0.78 | 0.50) (0.78 | 0.52)
                          Forced (0.67 | 0.17) (0.75 | 0.17)
                          Crosswise (0.17 | 0.00) (0.20 | 0.00) (0.23 | 0.00)
(0.25 | 0.00) (0.26 | 0.00) (0.30 | 0.00) (0.75 | 0.00) (0.83 | 0.00)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.28679    0.047932 1427
        DQ          0.21806    0.015389  720
    Forced          0.41265    0.017797 1417
       UQM          0.17351    0.022187  717

---------------------------------------------------------
Item:                           crib notes
Model(s):                  DQ (1.00 | 0.00)
                          UQM (0.70 | 0.49) (0.70 | 0.50) (0.70 | 0.52)
                              (0.78 | 0.49) (0.78 | 0.50) (0.78 | 0.52)
                          Forced (0.67 | 0.17) (0.75 | 0.17)
                          Crosswise (0.17 | 0.00) (0.20 | 0.00) (0.23 | 0.00)
(0.25 | 0.00) (0.26 | 0.00) (0.30 | 0.00) (0.75 | 0.00) (0.83 | 0.00)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.17570    0.044884 1427
        DQ          0.10278    0.011317  720
    Forced          0.32396    0.016822 1417
       UQM          0.14438    0.021528  717

---------------------------------------------------------
Item:                           drugs
Model(s):                  DQ (1.00 | 0.00)
                          UQM (0.70 | 0.49) (0.70 | 0.50) (0.70 | 0.52)
                              (0.78 | 0.49) (0.78 | 0.50) (0.78 | 0.52)
                          Forced (0.67 | 0.17) (0.75 | 0.17)
                          Crosswise (0.17 | 0.00) (0.20 | 0.00) (0.23 | 0.00)
```

```
(0.25 | 0.00) (0.26 | 0.00) (0.30 | 0.00) (0.75 | 0.00) (0.83 | 0.00)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.095850   0.0475537 1427
        DQ          0.029167   0.0062712  720
    Forced          0.162119   0.0138931 1417
       UQM          0.045145   0.0187692  717

----------------------------------------------------------
Item:                        handed in plagiarism
Model(s):                  DQ (1.00 | 0.00)
                          UQM (0.70 | 0.49) (0.70 | 0.50) (0.70 | 0.52)
                              (0.78 | 0.49) (0.78 | 0.50) (0.78 | 0.52)
                          Forced (0.67 | 0.17) (0.75 | 0.17)
                          Crosswise (0.17 | 0.00) (0.20 | 0.00) (0.23 | 0.00)
(0.25 | 0.00) (0.26 | 0.00) (0.30 | 0.00) (0.75 | 0.00) (0.83 | 0.00)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.081899   0.0444126 1427
        DQ          0.029167   0.0062712  720
    Forced          0.192582   0.0145974 1417
       UQM          0.070978   0.0195603  717

----------------------------------------------------------
Item:                        handed in someone else's work
Model(s):                  DQ (1.00 | 0.00)
                          UQM (0.70 | 0.49) (0.70 | 0.50) (0.70 | 0.52)
                              (0.78 | 0.49) (0.78 | 0.50) (0.78 | 0.52)
                          Forced (0.67 | 0.17) (0.75 | 0.17)
                          Crosswise (0.17 | 0.00) (0.20 | 0.00) (0.23 | 0.00)
 (0.25 | 0.00) (0.26 | 0.00) (0.30 | 0.00) (0.75 | 0.00) (0.83 | 0.00)

## Estimated Population Prevalence (weighted per RR model)
   RRmodel estimate.weighted se.weighted    n
 Crosswise          0.033014   0.0468658 1427
        DQ          0.015278   0.0045711  720
    Forced          0.146240   0.0134956 1417
       UQM          0.019240   0.0178254  717
```

The prevalence estimates are almost always the lowest for DQ. Most likely, students underreported their behavior when directly asked. In Figure 2, the weighted prevalence estimates are plotted for each item and RRT. This is the first of four plots when plotting an object of class `RRglmerMod` of the package **GLMMRR**. It can be seen that copying work is the most popular way, and then using a crib note. Although the CW condition has the highest number of students, the confidence interval of the CW estimates is much wider than for the other conditions. The CW method is less efficient in estimating student prevalence than the other RRTs.

Model fit

The relevance of the random person component is examined by comparing the multiple-group IRT model with a (multiple-group) GLM, which has the same fixed effect part.

```
R> out.fe <- RRglm(RR_response ~ 1 + Question + expcond, item=Question,
+      link = "RRlink.probit",RRmodel = RRmodel,p1=p1,p2=p2,data = ETHBE)
R> anova(out.re,out.fe)
Data: df
Models:
out.fe: RR_response ~ 1 + Question + expcond
out.re: RR_response ~ 1 + Question + expcond + (1 | id)
       npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
out.fe   10 21147 21227 -10564    21127
out.re   11 21016 21103 -10497    20994 133.82  1  < 2.2e-16 ***
```

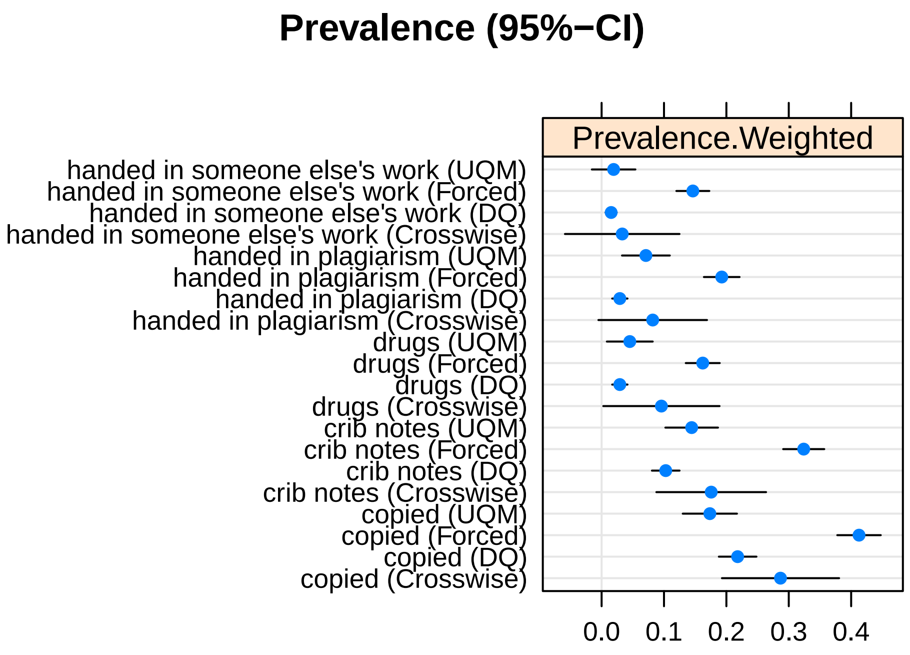


Figure 2: Prevalence estimates of types of student misconduct for different RRTs.

It follows that the RIRT model fits the data better than the GLM, with a lower AIC and BIC and a significant decrease in deviance. For the RIRT model, conditional Pearson residuals are computed given the random effect. The Pearson residuals are plotted against the fitted probabilities. To obtain the fitted probabilities, the fitted values on the linear predictor scale are computed using the `predict` function. Then, according to Equation 1, the fitted probabilities are computed using parameters c and d, which can be computed from function `getRRparameters` given the RRT and RR design parameters.

```
R> eta <- predict(out.re, type = "link")
R> dum <- getRRparameters(ETHBE$RRmodel, ETHBE$p1, ETHBE$p2)
R> pp <- dum$c + dum$d*pnorm(eta)
R> resid <- residuals(out.re, type = c("pearson"))
R> plot(pp[ETHBE$expcond=="direct questioning"],
+   resid[ETHBE$expcond=="direct questioning"],bty="l",
+   xlim=c(0,1),ylim=c(-2,6),cex=.8,xlab="Fitted probability",ylab="Pearson residual")
R> points(pp[ETHBE$expcond=="CM pick-a-number"],
+   resid[ETHBE$expcond=="CM pick-a-number"],cex=.8,pch=19,col="grey80")
R> points(pp[ETHBE$expcond=="FR pick-a-number"],
+   resid[ETHBE$expcond=="FR pick-a-number"],cex=.8,pch=15,col="red")
R> points(pp[ETHBE$expcond=="FR random wheel"],
+   resid[ETHBE$expcond=="FR random wheel"],cex=.8,pch=16,col="blue")
R> points(pp[ETHBE$expcond=="CM unrelated question"],
    resid[ETHBE$expcond=="CM unrelated question"],cex=.8,pch=17,col="green")
R> points(pp[ETHBE$expcond=="UQ Benford"],
+    resid[ETHBE$expcond=="UQ Benford"],cex=.8,pch=18,col="purple")
R> abline(h=1,lty=2,col="grey")
R> abline(h=-1,lty=2,col="grey")
R> legend(.6,6,c("DQ","FR pick-a-number","FR random wheel",
+   "CM unrelated", "UQ","CM pick-a-number"),
+   col=c("black","red","blue","green","purple","grey80"),
+   pch = c(1,15,16,17,18,19),cex=.7,bty="n")
```

In Figure 3, the Pearson residuals are plotted for each RRT. It can be seen that for DQ, the residuals are large (small) for positive (zero) responses since the prevalence rates are low. The residuals for FR and UQ are relatively small, partly because the corresponding fitted probabilities are in the middle of the scale. For the CW methods, it can be seen that the residuals are large (small) for zero (positive) responses since they correspond to high prevalence rates. Further analysis to improve the model by incorporating predictor variables is omitted for reasons of brevity.

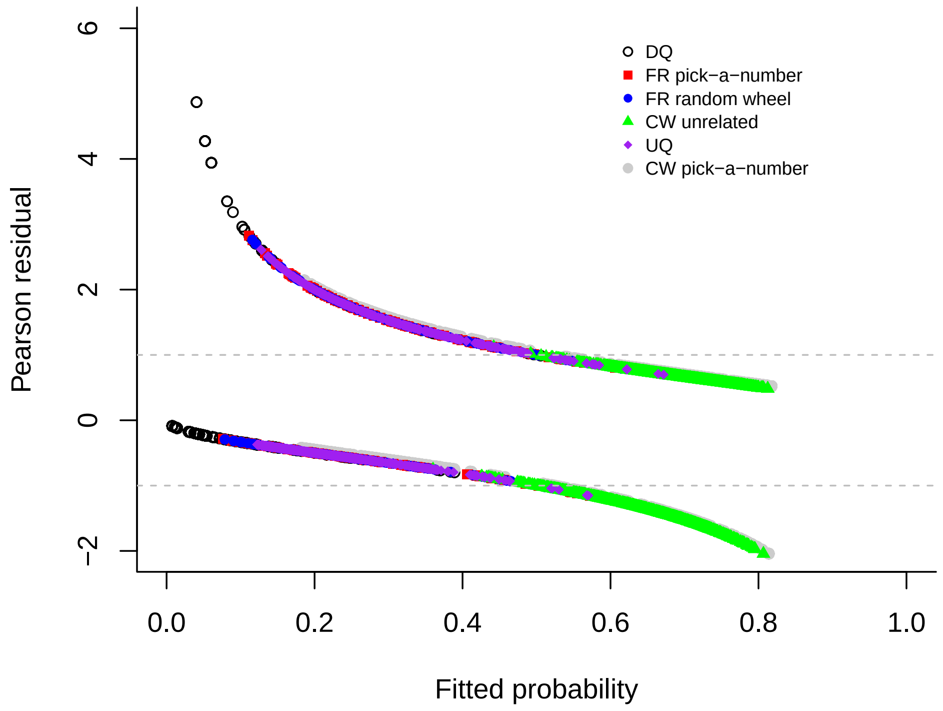


Figure 3: Fitted probabilities of types of student misconduct for different RRTs versus the Pearson residuals.

Discussion

The **GLMMRR** provides tools for fitting GLM(M)s on RR data, where the RR design and the design parameters are allowed to vary across observations. The multiple-group RR designs are particularly interesting to validate RR methods, to examine the sensitivity of the attributes, and to examine the influence of different levels of privacy protection. The multiple group modeling approach also supports simultaneously testing RR design effects across items and participants. The tools provide support to substantive applications using RR techniques, which can include different RR designs and advanced GLM and GLMM methods to analyze RR data. It is our objective to stimulate applied and methodological RR research by offering the open-source software **GLMMRR**. The tools extend the popular modeling tools of **lme4**, and class functions are simply extended to deal with RR data while maintaining the general features included in the GLM and GLMM software (e.g., **lme4**; Bates et al., 2015).

When some respondents do not follow the RR design instructions, the GLM(M) does not fit the data (e.g., Böckenholt and van der Heijden, 2007; Fox et al., 2013; De Jong et al., 2010). In that case, a composite link function can be used to include a linear predictor for those not following the instructions and one for those following the instructions. Currently, the ML estimations methods for **GLMMRR** cannot handle composite link functions. More research is needed to develop and implement estimation methods that can handle in a flexible way GLMMs with composite link functions (Thompson and Baker, 1981).

The modified link functions can be used for other types of GLMMs. In longitudinal research, in practice, time is often observed in discrete units. The discrete-time hazard defines the probability of the occurrence of an event at time t. The GLM(M) can be used to describe the link between the hazard rate and a linear predictor. The GLM(M) with the complementary log-log link function applies when the data is generated by a continuous-time proportional hazards model (Allison, 1982). Thus, for instance, when using an RR design to collect information about sensitive events (e.g., events related to war, trauma, sexual assault), the GLMM with a modified link function is an appropriate model to analyze the RR data. For count RR data, the complementary log-log link function can be used to model the probability of an RR of a non-zero observation (Fox et al., 2018). Ordinal data can be modeled with a (cumulative) link function for the proportional odds, which is defined by cumulative probabilities. For each response category, a GLMM defines a cumulative probability that response falls below or in this category. The modified link functions can be used to model cumulative probabilities with GLMMs for ordinal RR data.

The GLMMs can be computationally intensive and usually require relatively large sample sizes. RR designs also require larger samples sizes to achieve the same level of accuracy as DQ. Furthermore, the prevalence of sensitive behaviors is often relatively low, and to obtain reliable estimates, more data is required. The **GLMMRR** package supports large sample sizes, which can be around 10^6 observations.

Bibliography

P. D. Allison. Discrete-time methods for the analysis of event histories. *Sociological methodology*, 13: 61–98, 1982. URL https://doi.org/10.2307/270718. [p620]

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. URL https://doi.org/10.18637/jss.v067.i01. [p620]

G. Blair, Y.-Y. Zhou, and K. Imai. rr: Statistical methods for the randomized response technique, 2015. URL https://cran.r-project.org/package=rr. R package version 1.4. [p600]

U. Böckenholt and P. G. van der Heijden. Item randomized-response models for measuring noncompliance: Risk-return perceptions, social influences, and self-protective responses. *Psychometrika*, 72(2):245–262, 2007. URL https://doi.org/10.1007/s11336-005-1495-y. [p620]

R. F. Boruch. Maintaining confidentiality on data in educational research: A systemic analysis. *American Psychologist*, 26(5):413–430, 1971. URL https://doi.org/10.1037/h0031502. [p602]

J. M. Chambers and T. J. Hastie. *Statistical models in S*. Pacific Grove, CA: Wadsworth & Brooks, 1992. [p604]

P. De Boeck, M. Bakker, R. Zwitser, M. Nivard, A. Hofman, F. Tuerlinckx, I. Partchev, et al. The estimation of item response models with the lmer function from the lme4 package in R. *Journal of Statistical Software*, 39(12):1–28, 2011. URL https://doi.org/10.18637/jss.v039.i12. [p605]

M. G. De Jong, R. Pieters, and J.-P. Fox. Reducing social desirability bias through item randomized response: An application to measure underreported desires. *Journal of Marketing Research*, 47(1): 14–27, 2010. URL https://doi.org/10.1509/jmkr.47.1.14. [p620]

J.-P. Fox. Multilevel IRT using dichotomous and polytomous response data. *British Journal of Mathematical and Statistical Psychology*, 58(1):145–172, 2005. URL https://doi.org/10.1348/000711005X38951. [p600]

J.-P. Fox. Bayesian randomized item response theory models for sensitive measurement. In *Handbook of modern item response theory. Vol. 1. Models.* Chapman and Hall/CRC Press, 2012. [p600]

J.-P. Fox and R. R. Meijer. Using item response theory to obtain individual information from randomized response data: An application using cheating data. *Applied Psychological Measurement*, 32(8):595–610, 2008. URL https://doi.org/10.1177/0146621607312277. [p600]

J.-P. Fox, M. Avetisyan, and J. van der Palen. Mixture randomized item-response modeling: A smoking behavior validation study. *Statistics in medicine*, 32(27):4821–4837, 2013. URL https://doi.org/10.1002/sim.5859. [p600, 620]

J.-P. Fox, D. Veen, and K. Klotzke. Generalized linear mixed models for randomized responses. *Methodology*, 2018. URL https://doi.org/10.1027/1614-2241/a000153. [p602, 604, 621]

D. W. Heck and M. Moshagen. RRreg: An R package for correlation and regression analyses of randomized response data. *Journal of Statistical Software*, 85(2):1–29, 2018. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v085.i02. [p601]

M. Höglinger and B. Jann. More is not always better: An experimental individual-level validation of the randomized response technique and the crosswise model. *PloS one*, 13(8):e0201770, 2018. URL `https://doi.org/10.1371/journal.pone.0201770`. [p601, 602, 609, 611]

M. Höglinger, B. Jann, and A. Diekmann. Online survey on exams and written papers; documentation. University of Bern Social Sciences Working Papers 8, University of Bern, Department of Social Sciences, May 2014. URL `https://ideas.repec.org/p/bss/wpaper/8.html`. [p614, 617]

M. Höglinger, B. Jann, and A. Diekmann. Sensitive questions in online surveys: An experimental evaluation of different implementations of the randomized response technique and the crosswise model. In *Survey Research Methods*, volume 10, pages 171–187, 2016. URL `https://doi.org/10.18148/srm/2016.v10i3.6703`. [p601]

D. W. Hosmer and S. Lemeshow. Goodness of fit tests for the multiple logistic regression model. *Communications in statistics-Theory and Methods*, 9(10):1043–1069, 1980. URL `https://doi.org/10.1080/03610928008827941`. [p606]

L. K. John, G. Loewenstein, A. Acquisti, and J. Vosgerau. When and why randomized response techniques (fail to) elicit the truth. *Organizational Behavior and Human Decision Processes*, 148: 101–123, 2018. URL `https://doi.org/10.1016/j.obhdp.2018.07.004`. [p601]

A. Y. Kuk. Asking sensitive questions indirectly. *Biometrika*, 77(2):436–438, 1990. URL `https://doi.org/10.1093/biomet/77.2.436`. [p602]

G. J. Lensvelt-Mulders, J. J. Hox, P. G. van der Heijden, and C. J. Maas. Meta-analysis of randomized response research: Thirty-five years of validation. *Sociological Methods & Research*, 33(3):319–348, 2005. URL `https://doi.org/10.1177/0049124104268664`. [p600]

P. McCullagh and J. Nelder. *Generalized linear models (2nd ed.)*. London, UK: Chapman & Hall, 1989. [p603]

F. Rijmen, F. Tuerlinckx, P. De Boeck, and P. Kuppens. A nonlinear mixed model framework for item response theory. *Psychological methods*, 8(2):185–205, 2003. URL `https://doi.org/10.1037/1082-989X.8.2.185`. [p605]

N. Scheers and C. M. Dayton. Covariate randomized response models. *Journal of the American Statistical Association*, 83(404):969–974, 1988. URL `https://doi.org/10.1080/01621459.1988.10478686`. [p600]

R. Thompson and R. Baker. Composite link functions in generalized linear models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 30(2):125–131, 1981. URL `https://doi.org/10.2307/2346381`. [p620]

R. Tourangeau and T. W. Smith. Asking sensitive questions: The impact of data collection mode, question format, and question context. *The Public Opinion Quarterly*, 60(2):275–304, 1996. URL `http://www.jstor.org/stable/2749691`. [p600]

R. Tourangeau and T. Yan. Sensitive questions in surveys. *Psychological Bulletin*, 135(5):859–883, 2007. URL `https://doi.org/10.1037/0033-2909.133.5.859`. [p600]

G. Tutz. *Regression for categorical data*, volume 34. Cambridge University Press, 2011. [p603]

A. van den Hout, U. Böckenholt, and P. G. van der Heijden. Estimating the prevalence of sensitive behaviour and cheating with a dual design for direct questioning and randomized response. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(4):723–736, 2010. URL `https://doi.org/10.1111/j.1467-9876.2010.00720.x`. [p600]

S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. URL `https://doi.org/10.1080/01621459.1965.10480775`. [p600, 602]

F. Wolter and P. Preisendörfer. Asking sensitive questions: An evaluation of the randomized response technique versus direct questioning using individual validation data. *Sociological Methods & Research*, 42(3):321–353, 2013. URL `https://doi.org/10.1177/0049124113500474`. [p601]

J.-W. Yu, G.-L. Tian, and M.-L. Tang. Two new models for survey sampling with sensitive characteristic: design and analysis. *Metrika*, 67(3):251, 2008. URL `https://doi.org/10.1007/S00184-007-0131-X`. [p602]

Jean-Paul Fox
Department of Research Methodology, Measurement and Data Analysis
Faculty of Behavioural, Management and Social Sciences
University of Twente
7500AE Enschede, The Netherlands
E-mail: j.p.fox@utwente.nl

Konrad Klotzke
Department of Research Methodology, Measurement and Data Analysis
Faculty of Behavioural, Management and Social Sciences
University of Twente
7500AE Enschede, The Netherlands
E-mail: k.k.klotzke@utwente.nl

Duco Veen
Julius Global Health
Julius Center for Health Sciences and Primary Care
University Medical Center Utrecht
3584CX Utrecht, The Netherlands
Optentia Research Program
North-West University
1900 Vanderbijlpark, South Africa
E-mail: D.Veen-5@umcutrecht.nl

Visual Diagnostics for Constrained Optimisation with Application to Guided Tours

by H. Sherry Zhang, Dianne Cook, Ursula Laa, Nicolas Langrené, and Patricia Menéndez

Abstract A guided tour helps to visualise high-dimensional data by showing low-dimensional projections along a projection pursuit optimisation path. Projection pursuit is a generalisation of principal component analysis in the sense that different indexes are used to define the interestingness of the projected data. While much work has been done in developing new indexes in the literature, less has been done on understanding the optimisation. Index functions can be noisy, might have multiple local maxima as well as an optimal maximum, and are constrained to generate orthonormal projection frames, which complicates the optimization. In addition, projection pursuit is primarily used for exploratory data analysis, and finding the local maxima is also useful. The guided tour is especially useful for exploration because it conducts geodesic interpolation connecting steps in the optimisation and shows how the projected data changes as a maxima is approached. This work provides new visual diagnostics for examining a choice of optimisation procedure based on the provision of a new data object which collects information throughout the optimisation. It has helped to diagnose and fix several problems with projection pursuit guided tour. This work might be useful more broadly for diagnosing optimisers and comparing their performance. The diagnostics are implemented in the R package ferrn.

Introduction

Visualisation is widely used in exploratory data analysis (Tukey, 1977; Unwin, 2015; Healy, 2018; Wilke, 2019). Presenting information in graphics often unveils insights that would otherwise not be discovered and provides a more comprehensive understanding of the problem at hand. Task-specific tools such as Li et al. (2020) show how visualisation can be used to understand, for instance, the behaviour of the optimisation for the example of neural network classification models. However, no general visualisation tool is available for diagnosing optimisation procedures. The work presented in this paper brings visualization tools into optimisation problems with the aim to better understand the performance of optimisers in practice.

The focus of this paper is on the optimisation problem arising in the projection pursuit guided tour (Buja et al., 2005), an exploratory data analysis technique used for detecting interesting structures in high-dimensional data through a set of lower-dimensional projections (Cook et al., 2008). The goal of the optimisation is to identify the projection, represented by the projection matrix, that gives the most interesting low-dimensional view. A view is said to be interesting if it can show some structures of the data that depart from normality, such as bimodality, clustering, or outliers.

The optimization challenges encountered in the projection pursuit guided tour problem are common to those of optimization in general. Examples include the existence of multiple optima (local and global), the trade-off between computational burden and proximity to the optima, or dealing with noisy objective functions that might be non-smooth and non-differentiable (Jones et al., 1998). The visualization tools, optimization methods, and conceptual framework presented in this paper can therefore be applied to other optimization problems.

The remainder of the paper is organised as follows. The next section provides an overview of optimisation methods, specifically random search and line search methods. A review of the projection pursuit guided tour, an overview of the optimisation problem and, outlines of three existing algorithms follows. The third section presents the new visual diagnostics, including the design of a data structure to capture information during the optimisation, from which several diagnostic plots are created. An illustration of how the diagnostic plots can be used to examine the performance of different optimisers and guide improvements to existing algorithms is shown using simulated data. Finally, an explanation of the implementation in the R package, **ferrn** (Zhang et al., 2021), is provided.

Optimisation methods

The type of optimisation problem considered in this paper is constrained optimization (Bertsekas, 2014), assuming it is not possible to find a solution to the problem in the way of a closed-form. That is, the problem consists in finding the minimum or maximum of a function $f \in L^p$ in the constrained space $\mathcal{A}$, where L^p defines the vector space of function f, whose pth power is integrable.

Gradient-based methods are commonly used to optimise an objective function, with the most notable one being the gradient ascent (descent) method. Although these methods are popular, they rely on the availability of the objective function derivatives. As will be shown in the next section, the independent variables in our optimisation problem are the entries of a projection matrix, and the computational time required to perform differentiation on a matrix could impede the rendering of tour animation. In addition, some objective functions rely on the empirical distribution of the data, which makes it in general not possible to get the gradient. Hence, gradient-based methods are not the focus of this paper, and consideration will be given to derivative-free methods.

Derivative-free methods (Conn et al., 2009; Rios and Sahinidis, 2013), which do not rely on the knowledge of the gradient, are more generally applicable. Derivative-free methods have been developed over the years, where the emphasis is on finding, in most cases, a near-optimal solution. Here we consider three derivative-free methods, two of which are random search methods: creeping random search and simulated annealing, and the other one is pseudo-derivative search.

Random search methods (Romeijn, 2009; Zabinsky, 2013; Andradóttir, 2015) have a random sampling component as part of their algorithms and have been shown to have the ability to optimise non-convex and non-smooth functions. The initial random search algorithm, pure random search (Brooks, 1958), draws candidate points from the entire space without using any information of the current position and updates the current position when an improvement on the objective function is made. As the dimension of the space becomes larger, sufficient sampling from the entire space would require a long time for convergence to occur, despite a guaranteed global convergence (Spall, 2005). Various algorithms have thus been developed to improve pure random search by either concentrating on a narrower sampling space or using a different updating mechanism. Creeping random search (White, 1971) is such a variation, where a candidate point is generated within a neighbourhood of the current point. This makes creeping random search faster to compute but global convergence is no longer guaranteed. On the other hand, simulated annealing (Kirkpatrick et al., 1983; Bertsimas and Tsitsiklis, 1993), introduces a different updating mechanism. Rather than only updating the current point when an improvement is made, it uses a Metropolis acceptance criterion, where worse candidates still have a chance to be accepted. The convergence of simulated annealing algorithms has been widely researched (Mitra et al., 1986; Granville et al., 1994) and the global optimum can be attained under mild regularity conditions.

The pseudo-derivative search uses a common search scheme in optimisation: line search. In line search methods, users are required to provide an initial estimate x_1 and, at each iteration, a search direction S_k and a step size α_k are generated. Then one moves on to the next point following $x_{k+1} = x_k + \alpha_k S_k$ and the process is repeated until the desired convergence is reached. In derivative-free methods, local information of the objective function is used to determine the search direction. The choice of step size also needs consideration, as inadequate step sizes might prevent the optimisation method from converging to an optimum. An ideal step size can be chosen by finding the value of $\alpha_k \in \mathbb{R}$ that maximises $f(x_k + \alpha_k S_k)$ with respect to α_k at each iteration.

Projection pursuit guided tour

A projection pursuit guided tour combines two different methods (projection pursuit and guided tour) to explore interesting features in a high-dimensional space. Projection pursuit, coined by Friedman and Tukey (1974), detects interesting structures (e.g., clustering, outliers, and skewness) in multivariate data via low-dimensional projections. Guided tour (Cook et al., 1995) is one variation of a broader class of data visualisation methods, tour (Buja et al., 2005), which displays high-dimensional data through a series of animated projections.

Let $\mathbf{X}_{n \times p}$ be the data matrix with n observations in p dimensions. A d-dimensional projection is a linear transformation from $\mathbb{R}^p$ into $\mathbb{R}^d$ defined as $\mathbf{Y} = \mathbf{X} \cdot \mathbf{A}$, where $\mathbf{Y}_{n \times d}$ is the projected data and $\mathbf{A}_{p \times d}$ is the projection matrix. We define $f : \mathbb{R}^{n \times d} \mapsto \mathbb{R}$ to be an index function that maps the projected data $\mathbf{Y}$ onto a scalar value. This is commonly known as the projection pursuit index function, or just index function, and is used to measure the "interestingness" of a given projection. An interesting projection shows structures that are non-normal since theoretical proofs from Diaconis and Freedman (1984) have shown that projections tend to be normal as n and p approach infinity under certain conditions. There have been many index functions proposed in the literature, here are a few examples: early indexes that can be categorised as measuring the L^2 distance between the projection and a normal distribution: Legendre index (Friedman and Tukey, 1974); Hermite index (Hall, 1989); natural Hermite index (Cook et al., 1993); chi-square index (Posse, 1995) for detecting spiral structure; LDA index (Lee et al., 2005) and PDA (Lee and Cook, 2010) index for supervised classification; kurtosis index

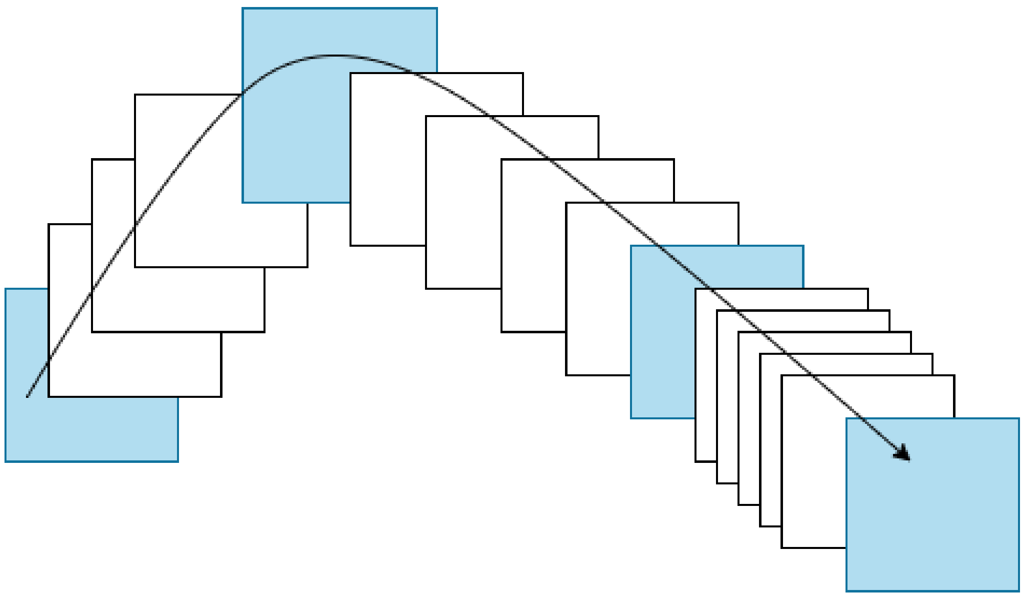


Figure 1: An illustration for demonstrating the frames in a tour path. Each square (frame) represents the projected data with a corresponding basis. Blue frames are returned by the projection pursuit optimisation and white frames are constructed between two blue frames by geodesic interpolation.

(Loperfido, 2020) and skewness index (Loperfido, 2018) for detecting outliers in financial time series; and most recently, scagnostic indexes (Laa and Cook, 2020) for summarising structures in scatterplot matrices based on eight scagnostic measures (Wilkinson et al., 2005; Wilkinson and Wills, 2008).

As a general visualisation method, tour produces animations of high-dimensional data via rotations of low-dimensional planes. There are different versions depending on how the high-dimensional space is investigated: grand tour (Cook et al., 2008) selects the planes randomly to provide a general overview; manual tour (Cook and Buja, 1997) gradually phases in and out one variable to understand the contribution of that variable in the projection. Guided tour, the main interest of this paper, chooses the planes with the aid of projection pursuit to gradually reveal the most interesting projection. Given a random start, projection pursuit iteratively finds bases with higher index values, and the guided tour constructs a geodesic interpolation between these planes to form a tour path. Figure @ref(fig:tour-path) shows a sketch of the tour path where the blue squares represent planes (targets) selected by the projection pursuit optimisation, and the white squares represent planes in the geodesic interpolation between targets. Mathematical details of the geodesic interpolation can be found in Buja et al. (2005). (Note that the term *frame* used in Buja's paper refers to a particular set of orthonormal vectors defining a plane. This is also conventionally referred to as a basis, which is used in this paper and the associated software.) The aforementioned tour method has been implemented in the R package **tourr** (Wickham et al., 2011).

Optimisation in the tour

In projection pursuit, the optimisation aims at finding the global and local maxima that give interesting projections according to an index function. That is, it starts with a given randomly selected basis $\mathbf{A}_1$ and aims at finding an optimal final projection basis $\mathbf{A}_T$ that satisfies the following optimisation problem:

$$\arg\max_{\mathbf{A}\in\mathcal{A}} f(\mathbf{X}\cdot\mathbf{A}) \quad s.t. \quad \mathbf{A}'\mathbf{A} = I_d\,, \tag{1}$$

where f and $\mathbf{X}$ are defined as in the previous section, $\mathcal{A}$ is the set of all p-dimensional projection bases, I_d is the d-dimensional identity matrix, and the constraint ensures the projection bases, $\mathbf{A}$, to be orthonormal. It is worth noticing the following: 1) The optimisation is constrained, and the orthonormality constraint imposes a geometrical structure on the bases space: it forms a Stiefel manifold. 2) There may be index functions for which the objective function might not be differentiable. 3) While finding the global optimum is the goal of the optimisation problem, interesting projections may also appear in the local optimum. 4) The optimisation should be fast to compute since the tour animation is viewed by the users during the optimisation.

Existing algorithms

Three optimisers have been implemented in the **tourr** (Wickham et al., 2011) package: creeping random search (CRS), simulated annealing (SA), and pseudo-derivative (PD). Creeping random search (CRS) is a random search optimiser that samples a candidate basis $\mathbf{A}_l$ in the neighbourhood of the current basis $\mathbf{A}_{\text{cur}}$ by $\mathbf{A}_l = (1-\alpha)\mathbf{A}_{\text{cur}} + \alpha\mathbf{A}_{\text{rand}}$ where $\alpha \in [0,1]$ controls the radius of the sampling neighbourhood and $\mathbf{A}_{\text{rand}}$ is generated randomly. $\mathbf{A}_l$ is then orthonormalised to fulfil the basis constraint. If $\mathbf{A}_l$ has an index value higher than the current basis $\mathbf{A}_{\text{cur}}$, the optimiser outputs $\mathbf{A}_l$ for a guided tour to construct an interpolation path. The neighbourhood parameter α is adjusted by a cooling parameter: $\alpha_{j+1} = \alpha_j * \text{cooling}$ before the next iteration starts. The optimiser terminates when the maximum number of iteration l_{max} is reached before a better basis can be found. The algorithm of CRS can be found in the appendix. Posse (1995) has proposed a slightly different cooling scheme by introducing a halving parameter c. In his proposal, α is only adjusted if the last iteration takes more than c times to find a better basis.

Simulated annealing (SA) uses the same sampling process as CRS but allows a probabilistic acceptance of a basis with lower index value than the current one. Given an initial value of $T_0 \in \mathbb{R}^+$, the "temperature" at iteration l is defined as $T(l) = \frac{T_0}{\log(l+1)}$. When a candidate basis fails to have an index value larger than the current basis, SA gives it a second chance to be accepted with probability

$$P = \min\left\{\exp\left[-\frac{\mid I_{\text{cur}} - I_l \mid}{T(l)}\right], 1\right\},$$

where $I_{(\cdot)} \in \mathbb{R}$ denotes the index value of a given basis. This implementation allows the optimiser to make a move and explore the basis space even if the candidate basis does not have a higher index value. Hence it enables the optimiser to jump out of a local optimum. The second algorithm in the appendix highlights how SA differs from CRS in the inner loop.

Pseudo-derivative (PD) search uses a different strategy than CRS and SA. Rather than randomly sample the basis space, PD first computes a search direction by evaluating bases close to the current basis. The step size is then chosen along the corresponding geodesic by another optimisation over a 90 degree angle from $-\pi/4$ to $\pi/4$. The resulting candidate basis $\mathbf{A}_{**}$ is returned for the current iteration if it has a higher index value than the current one. The third algorithm in the appendix summarises the inner loop of the PD.

Visual diagnostics

A data structure for diagnosing optimisers in projection pursuit guided tour is first defined. With this data structure, four types of diagnostic plots are presented.

Data structure for diagnostics

Three main pieces of information are recorded during the projection pursuit optimisation: 1) projection bases $\mathbf{A}$, 2) index values I, and 3) state S. For CRS and SA, possible states include `random_search`, `new_basis`, and `interpolation`. Pseudo-derivative (PD) has a wider variety of states, including `new_basis`, `direction_search`, `best_direction_search`, `best_line_search`, and `interpolation`. Multiple iterators index the information collected at different levels: t is a unique identifier prescribing the natural ordering of each observation; j and l are the counter of the outer and inner loop, respectively. Other parameters of interest recorded, V, include `method` that tags the name of the optimiser, and `alpha` that indicates the sampling neighbourhood size for searching observations. A matrix notation describing the data structure is:

t	$\mathbf{A}$	I	S	j	l	V_1	V_2	Description
1	$\mathbf{A}_1$	I_1	S_1	1	1	V_{11}	V_{12}	start basis
2	$\mathbf{A}_2$	I_2	S_2	2	1	V_{21}	V_{22}	search
...	...	...	...	...	...	...	...	...
...	...	...	...	2	l_2	...	...	search (accepted)
...	...	...	...	2	1	...	...	interpolation
...	...	...	...	...	...	...	...	...
...	...	...	...	2	k_2	...	...	interpolation
...	...	...	...	...	...	...	...	...
...	...	...	...	J	1	...	...	search
...	...	...	...	...	...	...	...	...
T	$\mathbf{A}_T$	I_T	S_T	J	l_J	V_{T1}	V_{T2}	search (final)
...	...	...	...	J	1	...	...	interpolation
...	...	...	...	...	...	...	...	...
...	...	...	...	J	k_J	...	...	interpolation
...	...	...	...	$J+1$	1	...	...	search (last round)
...	...	...	...	...	...	...	...	...
T'	$\mathbf{A}_{T'}$	$I_{T'}$	$S_{T'}$	$J+1$	l_{J+1}	$V_{T'1}$	$V_{T'2}$	search (last round)

where $T' = T + k_J + l_{J+1}$. Note that there is no output in iteration $J + 1$ since the optimiser does not find a better basis in the last iteration and terminates. The final basis found is A_T with index value I_T.

The data structure constructed above meets the tidy data principle (Wickham, 2014) that requires each observation to form a row and each variable to form a column. With tidy data structure, data wrangling and visualisation can be significantly simplified by well-developed packages such as **dplyr** (Wickham et al., 2020) and **ggplot2** (Wickham, 2016).

Diagnostic 1: Checking how hard the optimiser is working

A starting point of diagnosing an optimiser is to understand how many searches it has conducted, i.e., we want to summarise how the index is increasing over iterations and how many basis need to be sampled at each iteration. This is achieved using the function `explore_trace_search()`: a boxplot shows the distribution of index values for each try, where the accepted basis (corresponding to the highest index value) is always shown as a point. When there are only few tries at a given iteration, showing the data points directly is preferred over the boxplot and this is controlled via the `cutoff` argument. Additional annotations are added to facilitate better reading of the plot, and these include 1) the number of points searched in each iteration can be added as text label at the bottom of each iteration; 2) the anchor bases to interpolate are connected and highlighted in a larger size; 3) the colour of the last iteration is in greyscale to indicate no better basis found in this iteration.

Figure 2 shows an example of the search plot for CRS (left) and SA (right). Both optimisers quickly find better bases in the first few iterations and then take longer to find one in the later iterations. The anchor bases, the ones found with the highest index value in each iteration, always have an increased index value in the optimiser CRS while this is not the case for SA. This feature gives CRS an advantage in this simple example to quickly find the optimum.

Diagnostic 2: Examining the optimisation progress

Another interesting feature to examine is the changes in the index value between interpolating bases since the projection on these bases is shown in the tour animation. Trace plots are created by plotting the index value against time. Figure 3 presents the trace plot of the same optimisations as Figure 2, and one can observe that the trace is smooth in both cases. It may seem bizarre at first sight that the interpolation sometimes passes bases with higher index values before it decreases to a lower target. This happens because, on the one hand, the probabilistic acceptance in SA implies that some worse bases will be accepted by the optimiser. In addition, the guided tour interpolates between the current and target basis to provide a smooth transition between projections, and sometimes a higher index value will be observed along the interpolation path. This indicates that a non-monotonic interpolation cannot be avoided, even for CRS. Later, in Section *A problem of non-monotonicity*, there will be a discussion on improving the non-monotonic interpolation for CRS.

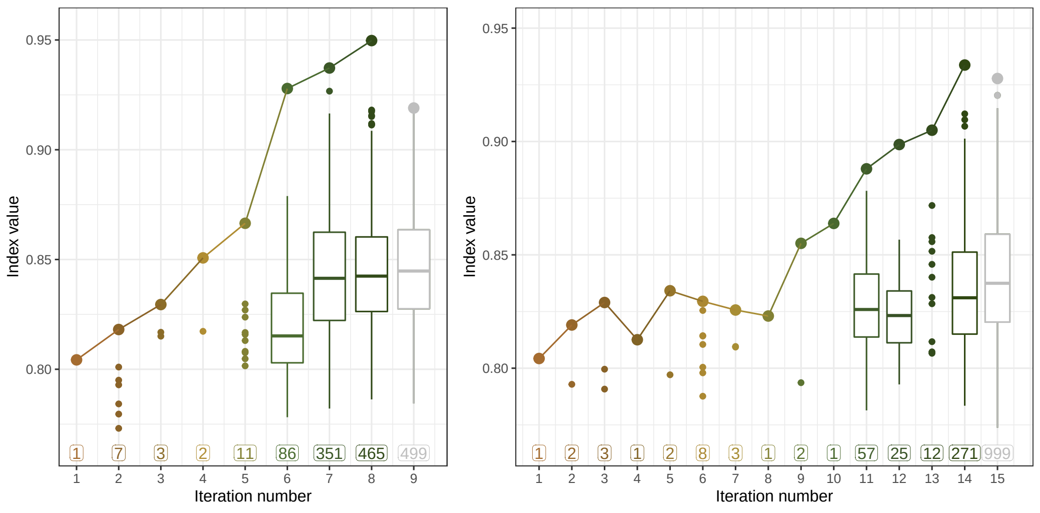


Figure 2: A comparison of the searches by two optimisers: CRS (left) and SA (right) on a 2D projection problem of a six-variable dataset, `boa6` using the holes index. Both optimisers reach the final basis with a similar index value, while it takes SA longer to find the final basis. In the earlier iterations, optimisers quickly find a better basis to proceed, while in the later iterations, most sampled bases fail to make an improvement on the index value, and a boxplot is used to summarise the distribution of the index values. There is no better basis found in the last iteration, 9 (left) and 15 (right), before reaching the maximum number of try and hence it is coloured grey. The colour scale is from the customised botanical palette in the **ferrn** package.

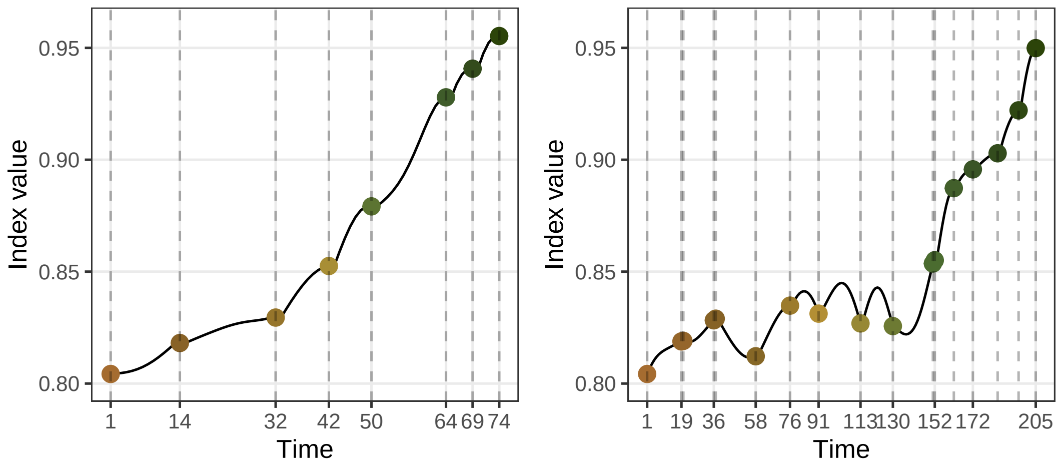


Figure 3: An inspection of the index values as the optimisation progress for two optimisers: CRS (left) and SA (right). The holes index is optimised for a 2D projection problem on the six-variable dataset `boa6`. Lines indicate the interpolation, and dots indicate new target bases generated by the optimisers. Interpolation in both optimisation is smooth, while SA is observed to first pass by some bases with higher index values before reaching the target bases in time 76-130.

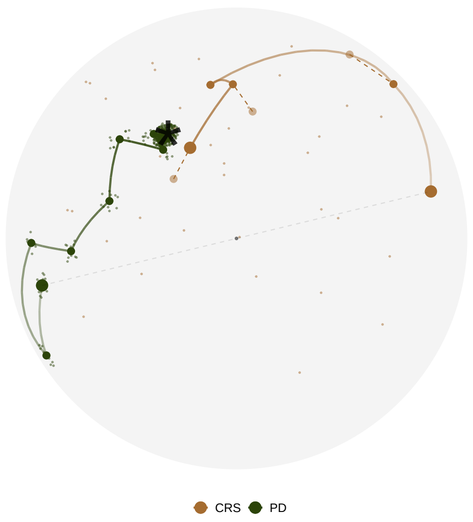


Figure 4: Search paths of CRS (green) and PD (brown) in the PCA-reduced basis space for 1D projection problem on the five-variable dataset, `boa5` using holes index. The basis space, a 5D unit sphere, is projected onto a 2D circle by PCA. The black star represents the theoretical best basis the optimisers are aiming to find. All the bases in PD have been flipped for easier comparison of the final bases, and a grey dashed line has been annotated to indicate the symmetry of the two start bases.

Diagnostic 3a: Understanding the optimiser's coverage of the search space

Apart from checking the search and progression of an optimiser, looking at where the bases are positioned in the basis space is also of interest. Given the orthonormality constraint, the space of projection bases $\mathbf{A}_{p\times d}$ is a Stiefel manifold. For one-dimensional projections, this forms a p-dimensional sphere. A dimensionality reduction method, e.g., principal component analysis, is applied to first project all the bases onto a 2D space. In a projection pursuit guided tour optimisation, there are various types of bases involved: 1) The starting basis; 2) The search bases that the optimiser evaluated to produce the anchor bases; 3) The anchor bases that have the highest index value in each iteration; 4) The interpolating bases on the interpolation path; and finally, 5) the end basis. The importance of these bases differs but the most important ones are the starting, interpolating, and end bases. Sometimes, two optimisers can start with the same basis but finish with bases of opposite signs. This happens because the projection is invariant to the orientation of the basis, and so is the index value. However, this creates difficulties for comparing the optimisers since the end bases will be symmetric to the origin. A sign flipping step is conducted to flip the signs of all the bases in one routine if different optimisations finish at opposite places.

Several annotations have been made to help understand this plot. As mentioned previously, the original basis space is a high-dimensional sphere, and random bases on the sphere can be generated via the **geozoo** (Schloerke, 2016) package. We use PCA to project and visualize the parameters/ bases in 2D. The centre of the 2D view is the first two PCs of the data matrix. It theoretically should be a circle but may have some irregular edges due to finite sampling. Thus the edge is smoothed by using a radius estimated as the largest distance from the centre to any basis. In the simulation, the theoretical best basis is known and can be labelled to compare how close to this that the optimisers stopped. Various aesthetics, i.e., size, alpha (transparency), and colour, are applicable to emphasize critical elements and adjust for the presentation. For example, anchor points and search points are less important, and hence a smaller size and alpha are used. Alpha can also be applied on the interpolation paths to show start to finish from transparent to opaque.

Figure 4 shows the PCA plot of CRS and PD for a 1D projection problem. Both optimisers find the optimum, but PD gets closer. With the PCA plot, one can visually appreciate the nature of these two optimisers: PD first evaluates points in a small neighbourhood for a promising direction, while CRS evaluates points randomly in the search space to search for the next target. There are dashed lines annotated for CRS, and it describes the interruption of the interpolation, which will be discussed in detail in Section *A problem of non-monotonicity*.

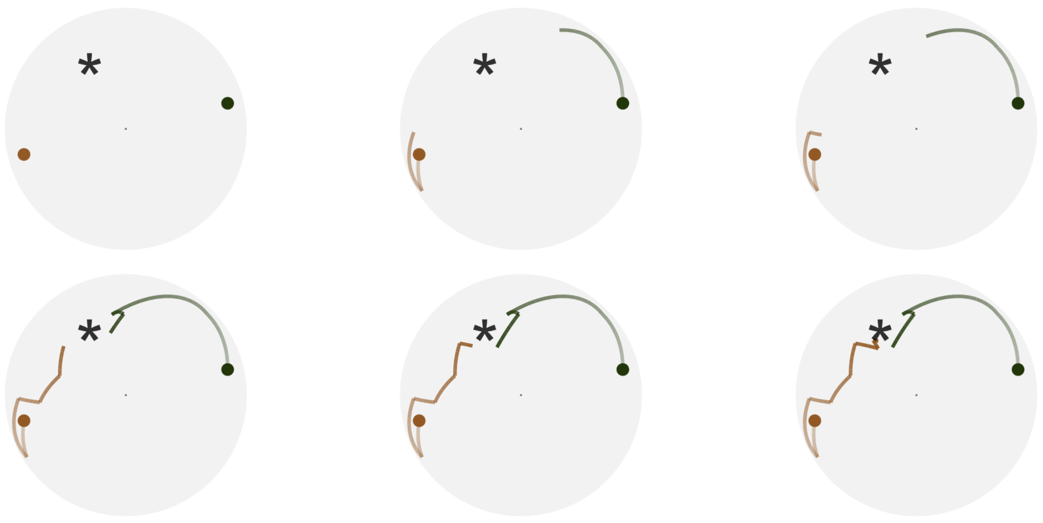


Figure 5: Six frames selected from the animated version of the previous plot. With animation, the progression of the search paths from start to finish is better identified. CRS (green) finishes the optimisation quicker than PD (brown) since there is no further movement for CRS in the sixth frame. The full video of the animation can be found in the html version of the paper.

Diagnostic 3b: Animating the diagnostic plots

Animation is another type of display to show how the search progresses from start to finish in the space. Figure 5 shows the animated version (six frames from the animation if viewed in pdf) of the PCA plot in Figure 4. An additional piece of information one can learn from this animation is that CRS finds its end basis quicker than PD since CRS finishes its search in the 5th frame while PD is still making more progress.

Diagnostic 4a: The tour looking at itself

As mentioned previously, the original $p \times d$ dimension space can be simulated via randomly generated bases in the **geozoo** (Schloerke, 2016) package. While the PCA plot projects the bases from the direction that maximises the variance, the tour plot displays the original high-dimensional space from various directions using animation. Figure 6 shows some frames from the tour plot of the same two optimisations in its original space.

Diagnostic 4b: Forming a torus

While the previous few examples have looked at the space of 1D basis in a unit sphere, this section visualises the space of 2D basis. Recall that the columns in a 2D basis are orthogonal to each other, so the space of $p \times 2$ bases is a torus in the p-D space (Buja and Asimov, 1986). For $p = 3$ one would see a classical 3D torus shape as shown by the grey points in Figure 7. The two circles of the torus can be observed to be perpendicular to each other and this can be linked back to the orthogonality condition. Two paths from CRS and PD are plotted on top of the torus and coloured in green and brown, respectively, to match the previous plots. The final basis found by PD and CRS are shown in a larger shape and printed below, respectively:

```
#>              [,1]        [,2]
#> [1,]  0.001196285  0.03273881
#> [2,] -0.242432715  0.96965761
#> [3,] -0.970167484 -0.24226493
```

```
#>             [,1]         [,2]
#> [1,]  0.05707994 -0.007220138
#> [2,] -0.40196202 -0.915510160
#> [3,] -0.91387549  0.402230054
```

Both optimisers have found the third variable in the first direction and the second variable in the second direction. Note, however, the different orientation of the basis, following from the different

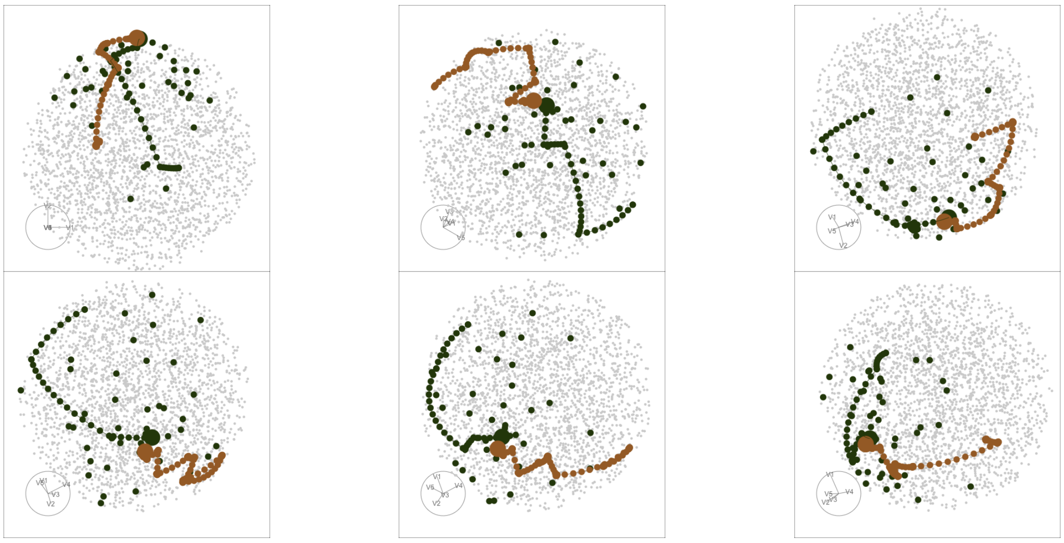


Figure 6: Six frames selected from rotating the high-dimensional basis space, along with the same two search paths from Figure 4 and 5. The basis space in this example is a 5D unit sphere, on which points (grey) are randomly generated via the CRAN package geozoo. The full animation can be seen in the html version of the paper.

sign in the second column. One would expect to see this in the torus plot as the final bases match each other when projected onto one torus circle (due to the same sign in the first column) and are symmetric when projected onto the other (due to the different sign in the second column). In Figure 7, this can be seen most clearly in frame 5 where the two circles are rotated into a line from our view.

Diagnosing an optimiser

In this section, several examples will be presented to show how the diagnostic plots discover something unexpected in projection pursuit optimisation, and guide the implementation of new features.

Simulation setup

Random variables with different distributions have been simulated as follows:

$$x_1 \overset{d}{=} x_8 \overset{d}{=} x_9 \overset{d}{=} x_{10} \sim \mathcal{N}(0,1) \tag{2}$$

$$x_2 \sim 0.5\mathcal{N}(-3,1) + 0.5\mathcal{N}(3,1) \tag{3}$$

$$\Pr(x_3) = \begin{cases} 0.5 & \text{if } x_3 = -1 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$x_4 \sim 0.25\mathcal{N}(-3,1) + 0.75\mathcal{N}(3,1) \tag{5}$$

$$x_5 \sim \frac{1}{3}\mathcal{N}(-5,1) + \frac{1}{3}\mathcal{N}(0,1) + \frac{1}{3}\mathcal{N}(5,1) \tag{6}$$

$$x_6 \sim 0.45\mathcal{N}(-5,1) + 0.1\mathcal{N}(0,1) + 0.45\mathcal{N}(5,1) \tag{7}$$

$$x_7 \sim 0.5\mathcal{N}(-5,1) + 0.5\mathcal{N}(5,1) \tag{8}$$

Variables `x1`, `x8` to `x10` are normal noise with zero mean, and unit variance and `x2` to `x7` are normal mixtures with varied weights and locations. All the variables have been scaled to have overall unit variance before projection pursuit. The holes index (Cook et al., 2008), used for detecting bimodality of the variables, is used throughout the examples unless otherwise specified.

A problem of non-monotonicity

An example of non-monotonic interpolation has been given in Figure 3: a path that passes bases with a higher index value than the target one. For SA, a non-monotonic interpolation is justified since target

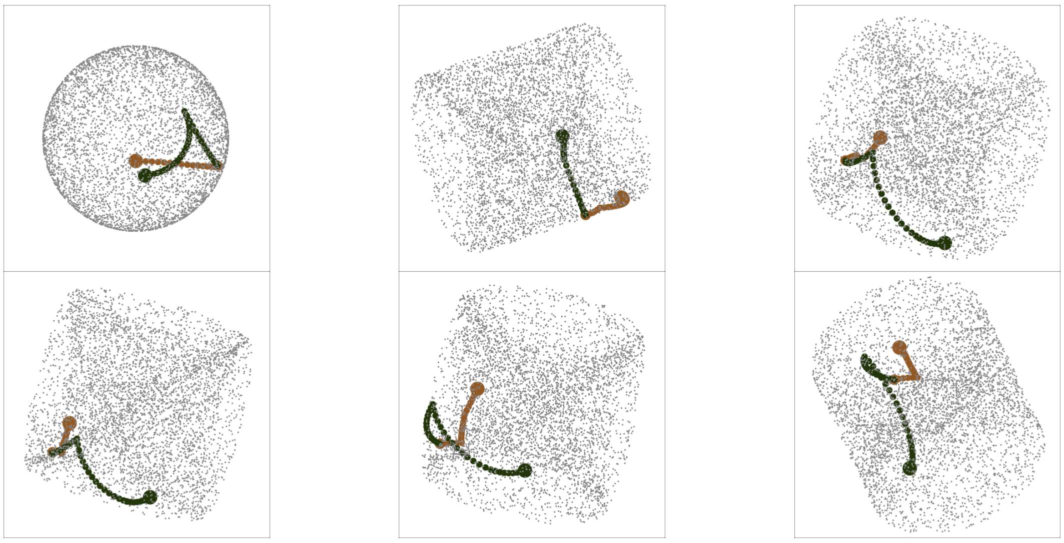


Figure 7: Six frames selected from rotating the 2D basis space along with two search paths optimised by PD (brown) and CRS (green). The projection problem is a 2D projection with three variables using the holes index. The grey points are randomly generated 2D projection bases in the space, and it can be observed that these points form a torus. The full video of the animation can be found in the html version of the paper.

bases do not necessarily have a higher index value than the current one, while this is not the case for CRS. The original trace plot for a 2D projection problem, optimised by CRS, is shown on the left panel of Figure 8, and one can observe that the non-monotonic interpolation has undermined the optimiser to realise its full potential. Hence, an interruption is implemented to stop at the best basis found in the interpolation. The right panel of Figure 8 shows the trace plot after implementing the interruption, and while the first two interpolations are identical, the basis at time 61 has a higher index value than the target in the third interpolation. Rather than starting the next iteration from the target basis on time 65, CRS starts the next iteration at time 61 on the right panel and reaches a better final basis.

Close but not close enough

Once the final basis has been found by an optimiser, one may want to push further in the close neighbourhood to see if an even better basis can be found. A polish search takes the final basis of an optimiser as the start of a new guided tour to search for local improvements. The polish algorithm is similar to the CRS but with three distinctions: 1) a hundred rather than one candidate bases are generated each time in the inner loop; 2) the neighbourhood size is reduced in the inner loop, rather than in the outer loop; and 3) three more termination conditions have been added to ensure the new basis generated is distinguishable from the current one in terms of the distance in the space, the relative change in the index value, and neighbourhood size:

1) the distance between the basis found and the current needs to be larger than 1e-3;
2) the relative change of the index value needs to be larger than 1e-5; and
3) the alpha parameter needs to be larger than 0.01.

Figure 9 presents the projected data and trace plot of a 2D projection, optimised by CRS and followed by the polish step. The top row shows the initial projection, the final projection after CRS, and the final projection after polish, respectively. The end basis found by CRS reveals the four clusters in the data, but the edges of each cluster are not clean-cut. Polish works with this end basis and further pushes the index value to produce clearer edges of the cluster, especially along the vertical axis.

Seeing the signal in the noise

The holes index function used for all the examples before this section produces a smooth interpolation, while this is not the case for all the indexes. An example of a noisy index function for 1D projections compares the projected data, $\mathbf{Y}_{n\times 1}$, to a randomly generated normal distribution, $\mathcal{N}_{n\times 1}$, using the Kolmogorov test. Let $F_{.}(n)$ be the empirical cumulative distribution function (ECDF) with two possible subscripts, Y and $\mathcal{N}$, representing the projected and randomly generated data, and n denoting the number of observations, the Kolmogorov index $I^{nk}(n)$, is defined as:

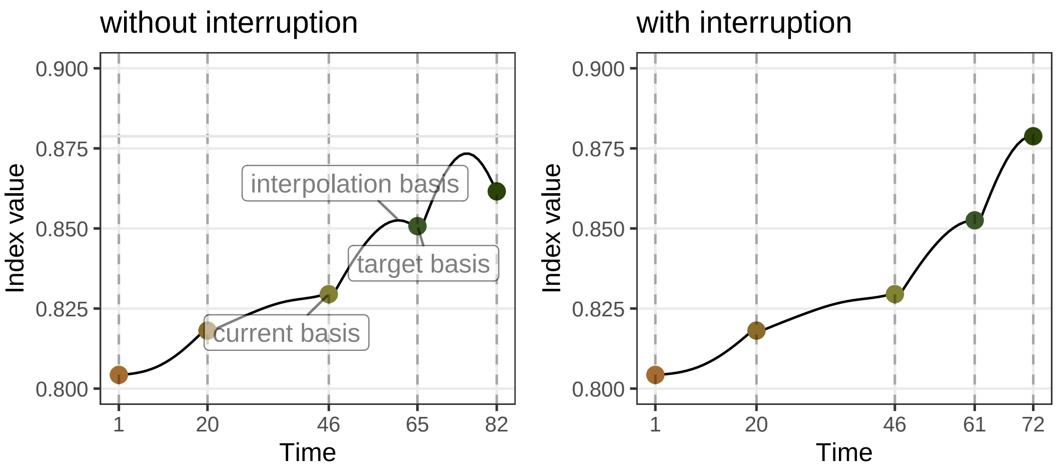


Figure 8: Comparison of the interpolation before and after implementing the interruption for the 2D projection problem on `boa6` data using holes index, optimised by CRS. On the left panel, the basis with a higher index value is found during the interpolation but not used. On the right panel, the interruption stops the interpolation at the basis with the highest index value for each iteration and results in a final basis with a higher index value, as shown on the right panel.

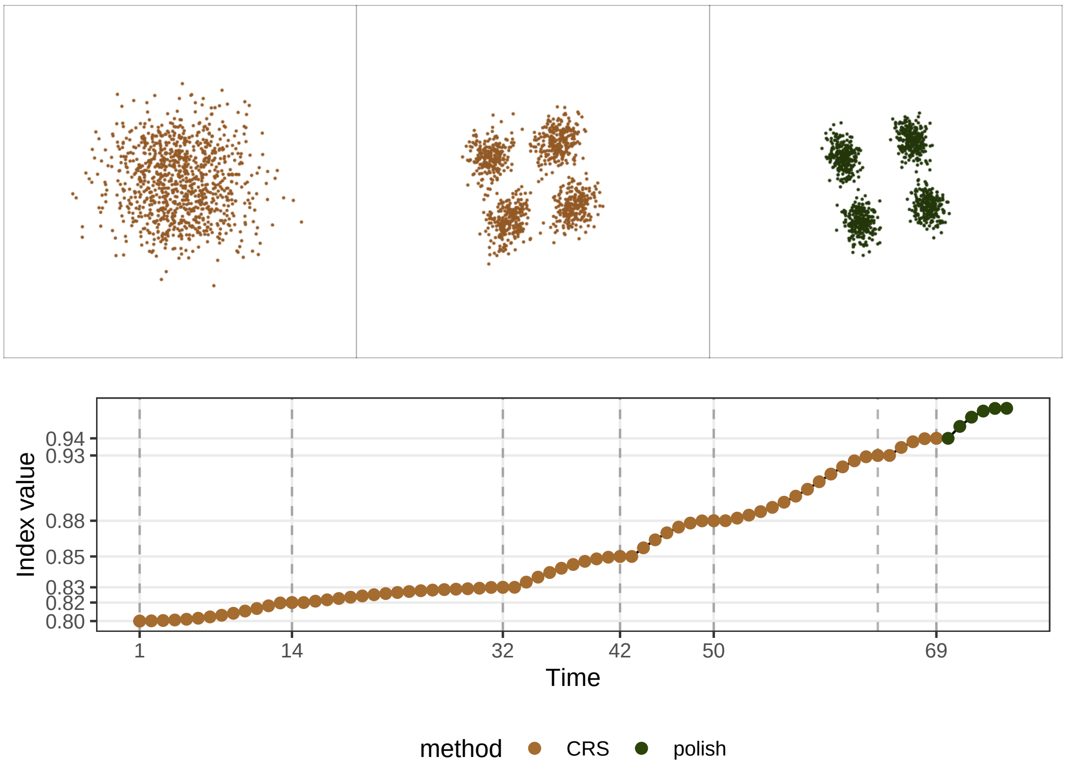


Figure 9: Comparison of the projected data before and after using polishing for a 2D projection problem on `boa6` data using holes index. The top row shows the initial projected data and the final views after CRS and polish search, and the second row traces the index value. The clustering structure in the data is detected by CRS (top middle panel), but the polish step improves the index value and produces clearer boundaries of the clusters (top right panel), especially along the vertical axis. Note that the parameter `max.tries` is set to 400 in this experiment for CRS to do its best.

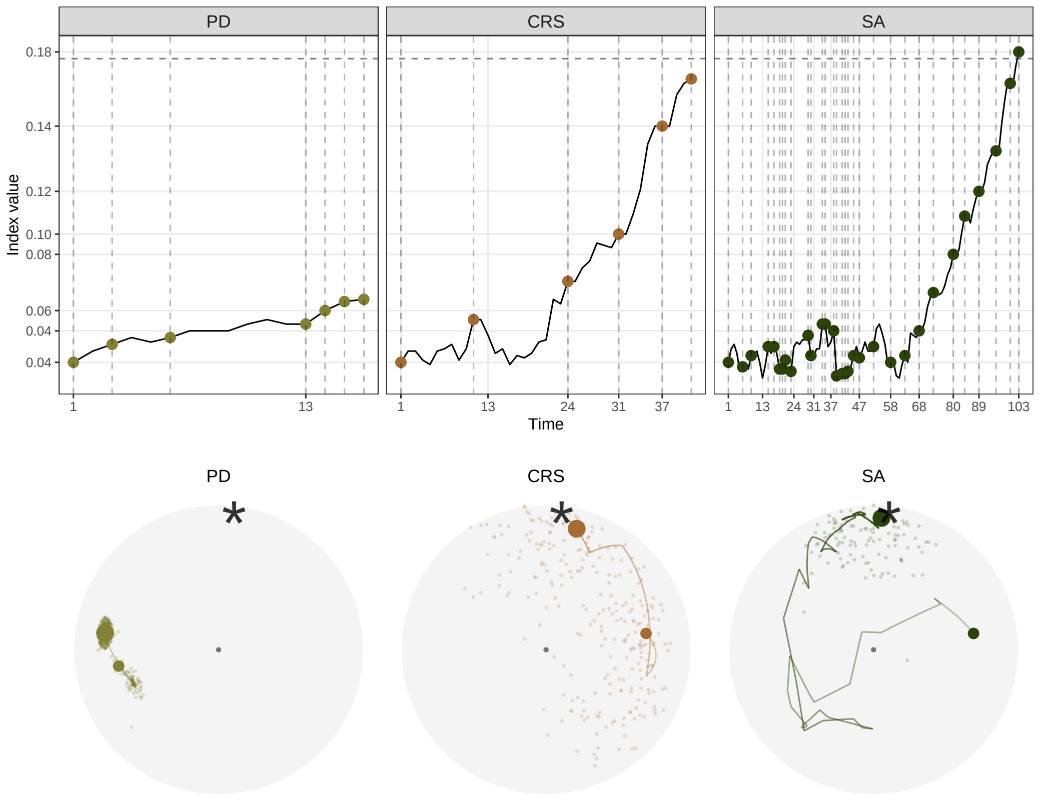


Figure 10: Comparison of the three optimisers in optimising $I^{nk}(n)$ index for a 1D projection problem on a five-variable dataset, boa5. Both CRS and SA succeed in the optimisation, PD fails to optimise this non-smooth index. Further, SA takes much longer than CRS to finish the optimisation, but finishes off closer to the theoretical best.

$$I^K(n) = \max\left[F_Y(n) - F_{\mathcal{N}}(n)\right].$$

With a non-smooth index function, two research questions are raised:

1) whether any optimiser fails to optimise this non-smooth index; and
2) whether the optimisers can find the global optimum despite the presence of a local optimum.

Figure 10 presents the trace and PCA plots of all three optimisers, and as expected, none of the interpolated paths are smooth. There is barely any improvement made by PD, indicating its failure in optimising non-smooth index functions. While CRS and SA have managed to get close to the index value of the theoretical best, the trace plot shows that it takes SA much longer to find the final basis. This long interpolation path is partially due to the fluctuation in the early iterations, where SA tends to generously accept inferior bases before concentrating near the optimum. The PCA plot shows the interpolation path and search points, excluding the last termination iteration. Pseudo-Derivative (PD) quickly gets stuck near the starting position. Comparing the amount of random search done by CRS and SA, it is surprising that SA does not carry as many samples as CRS. Combining the insights from both the trace and PCA plot, one can learn the two different search strategies by CRS and SA: CRS frequently samples in the space and only make a move when an improvement is guaranteed to be made, while SA first broadly accepts bases in the space and then starts the extensive sampling in a narrowed subspace. The specific decision of which optimiser to use will depend on the index curve in the basis space, but if the basis space is non-smooth, accepting inferior bases at first, as SA has done here, can lead to a more efficient search in terms of the overall number of points evaluated.

The next experiment compares the performance of CRS and SA when a local maximum exists. Two search neighbourhood sizes, 0.5 and 0.7, are compared to understand how a large search neighbourhood would affect the final basis and the length of the search. Figure 11 shows 80 paths simulated using 20 seeds in the PCA plot, faceted by the optimiser and search size. With CRS and a search size of 0.5, despite being the simplest and fastest, the optimiser fails in three instances where the final basis lands neither near the local nor the global optimum. With a larger search size of 0.7, more seeds have

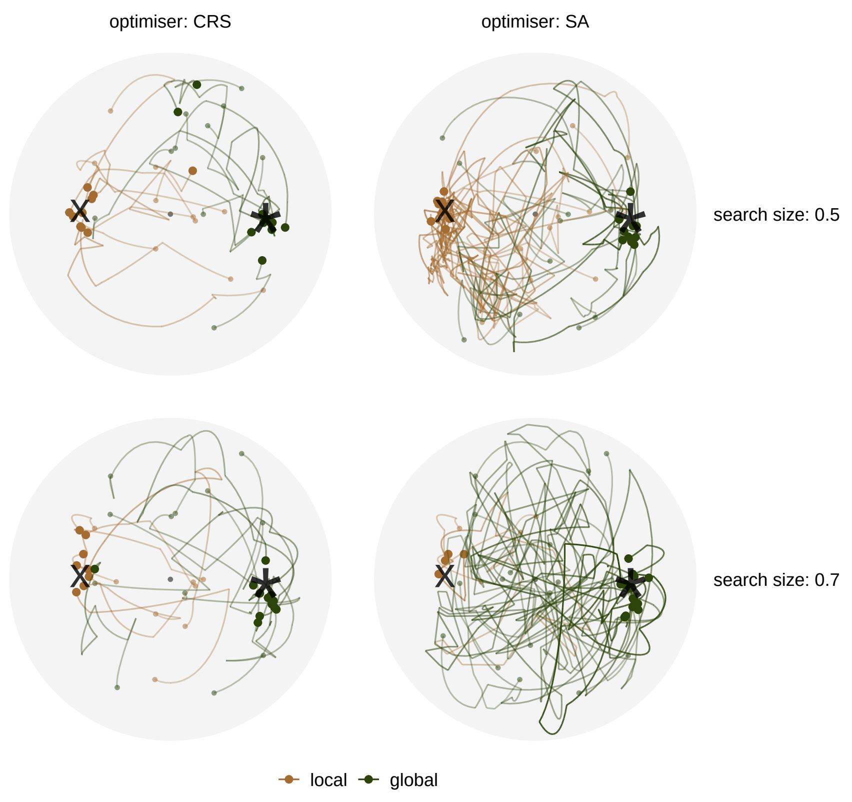


Figure 11: Comparing 20 search paths in the PCA-projected basis space faceted by two optimisers: CRS and SA, and two search sizes: 0.5 and 0.7. The optimisation is on the 1D projection index, $I^{nk}(n)$, for `boa6` data, where a local optimum, annotated by the cross (x), is presented in this experiment, along with the global optimum (*).

found the global maximum. Comparing CRS and SA for a search size of 0.5, SA does not seem to improve the final basis found, despite having longer interpolation paths. However, the denser paths near the local maximum are an indicator that SA is working hard to examine if there is any other optimum in the basis space, but the relatively small search size has diminished its ability to reach the global maximum. With a larger search size, almost all the seeds (16 out of 20) have found the global maximum, and some final bases are much closer to the theoretical best, as compared to the three other cases. This indicates that SA, with a reasonable large search window, is able to overcome the local optimum and optimise close towards the global optimum.

Reconciling the orientation

One interesting situation observed in the previous examples is that, for some simulations, as shown on the left panel of Figure 12, the target basis is generated on the other half of the basis space, and the interpolator seems to draw a straight line to interpolate. Bases with opposite signs do not affect the projection and index value, but we would prefer the target to have the same orientation as the current basis. The orientation of two bases can be computationally checked by calculating the determinant – a negative value suggests the two bases have a different orientation. For 1D bases, this can be corrected by flipping the sign on one basis. For higher dimensions, it can be a bit more difficult because the orthonormality of the basis needs to be also maintained when an individual vector is flipped. Here, an orientation check is carried out once a new target basis is generated, and the sign in the target basis will be flipped if a negative determinant is obtained. The interpolation after implementing the orientation check is shown on the right panel of Figure 12, where the unsatisfactory interpolation no longer exists.

Implementation

This project contributes to the software development in two packages: a data collection object is implemented in **tourr** (Wickham et al., 2011), while the visual diagnostics of the optimisers is implemented in **ferrn** (Zhang et al., 2021). The functions in the **ferrn** (Zhang et al., 2021) package are listed below:

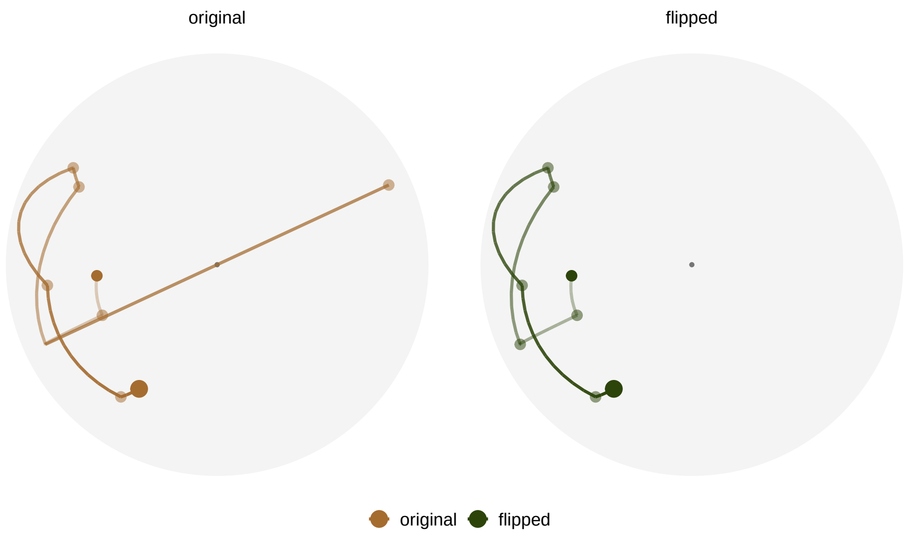


Figure 12: Comparison of the interpolation in the PCA-projected basis space before and after reconciling the orientation of the target basis. Optimisation is on the 1D projection index, $I^{nk}(n)$, for boa6 data using CRS with seed 2463. The dots represent the target basis in each iteration, and the path shows the interpolation. On the left panel, one target basis is generated with an opposite orientation to the current basis (hence appear on the other side of the basis space), and the interpolator crosses the origin to perform the interpolation. The right panel shows the same interpolation after implementing an orientation check, and the undesirable interpolation disappears.

- Main plotting functions:
  - `explore_trace_search()` produces summary plots in Figure 2.
  - `explore_trace_interp()` produces trace plots for the interpolation points in Figure 3.
  - `explore_space_pca()` produces the PCA plot of projection bases on the reduced space. Figure 4 includes the additional details of anchor and search bases, which can be turned on by the argument `details = TRUE`. The animated version in Figure 5 is produced with argument `animate = TRUE`.
  - `explore_space_tour()` produces animated tour view on the full space of the projection bases in Figure 6.
- `get_*()` extracts and manipulates certain components from the existing data object.
  - `get_anchor()` extracts target observations.
  - `get_basis_matrix()` flattens all the bases into a matrix.
  - `get_best()` extracts the observation with the highest index value in the data object.
  - `get_dir_search()` extracts directional search observations for PD search.
  - `get_interp()` extracts interpolated observations.
  - `get_interp_last()` extracts the ending interpolated observations in each iteration.
  - `get_interrupt()` extracts the ending interpolated observations and the target observations if the interpolation is .interrupted
  - `get_search()` extracts search observations.
  - `get_search_count()` extracts the count of search observations.
  - `get_space_param()` produces the coordinates of the centre and radius of the basis space.
  - `get_start()` extracts the starting observation.
  - `get_theo()` extracts the theoretical best observations, if given.
- `bind_*()` incorporates additional information outside the tour optimisation into the data object.
  - `bind_theoretical()` binds the theoretical best observation in simulated experiment.
  - `bind_random()` binds randomly generated bases in the projection bases space to the data object.
  - `bind_random_matrix()` binds randomly generated bases and outputs in a matrix format.
- `add_*()` provides wrapper functions to create ggprotos for different components for the PCA plot
  - `add_anchor()` for plotting anchor bases.
  - `add_anno()` for annotating the symmetry of start bases.

  - `add_dir_search()` for plotting the directional search bases with magnified distance.
  - `add_end()` for plotting end bases.
  - `add_interp()` for plotting the interpolation path.
  - `add_interp_last()` for plotting the last interpolation bases for comparing with target bases when interruption is used.
  - `add_interrupt()` for linking the last interpolation bases with target ones when interruption is used.
  - `add_search()` for plotting search bases.
  - `add_space()` for plotting the circular space.
  - `add_start()` for plotting start bases.
  - `add_theo()` for plotting theoretical best bases, if applicable.

- Utilities
  - `theme_fern()` and `format_label()` for better display of the grid lines and axis formatting.
  - `clean_method()` to clean up the name of the optimisers.
  - `botanical_palettes()` is a collection of colour palettes from Australian native plants. Quantitative palettes include daisy, banksia, and cherry, and sequential palettes contain fern and acacia.
  - `botanical_pal()` as the colour interpolator.
  - `scale_color_*()` and `scale_fill_*()` for scaling the colour and fill of the plot.

Conclusion

This paper has provided several visual diagnostics that can be used for understanding a complex optimisation procedure and are implemented in the **ferrn** package. The methods were illustrated using the optimisers available for projection pursuit guided tour. Here the constraint is the orthonormality condition of the projection bases, which corresponds to optimisation over spheres and torii. The approach described broadly applies to other constrained optimisers. Although the manifold in p-space might be different the diagnostic techniques are the same. A researcher would begin by saving the path of the optimiser in a form required to input into the ferrn package, as described in this paper. One might generally make more samples from the constrained space upon which to assess and compare the optimisation paths. These high-dimensional data objects can then be viewed using the tour.

The progressive optimisation of a target function and its coverage of the search space can be viewed in both reduced 2D space and the full space. These visualisations can lead to insights for evaluating and comparing the performance of multiple optimisers operating on the same task. They can provide a better understanding of existing methods or motivate the development of new approaches. For example, we have compared how three optimisers perform when maximising a non-smooth index function and have illustrated how the pseudo-derivative search fails in this setting. The observations from our experiments have also been translated into improved optimisation methods for the guided tour, e.g., we introduced the option to interrupt the search if a better basis is found along the path.

This work might be considered an effort to bring transparency into algorithms. Although little attention is paid by algorithm developers to providing ways to output information during intermediate steps, this is an important component for enabling others to understand and diagnose the performance. Algorithms are an essential component of artificial intelligence that is used to make daily life easier. Interpretability of algorithms is important to guard against aspects like bias and inappropriate use. The data object underlying the visual diagnostics here is an example of what might be useful in algorithm development generally.

Acknowledgements

This article is created using **knitr** (Xie, 2015) and **rmarkdown** (Xie et al., 2018) in R. The source code for reproducing this paper can be found at: `https://github.com/huizezhang-sherry/paper-ferrn`.

Bibliography

S. Andradóttir. A review of random search methods. In *Handbook of Simulation Optimization*, pages 277–292. Springer, 2015. URL `https://doi.org/10.1007/978-1-4939-1384-8`. [p625]

D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014. URL `https://doi.org/10.1016/C2013-0-10366-2`. [p624]

D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993. URL https://doi.org/10.1214/ss/1177011077. [p625]

S. H. Brooks. A discussion of random methods for seeking maxima. *Operations Research*, 6(2):244–251, 1958. URL https://doi.org/10.1287/opre.6.2.244. [p625]

A. Buja and D. Asimov. Grand tour methods: an outline. In *Proceedings of the Seventeenth Symposium on the Interface of Computer Sciences and Statistics*, pages 63–67, USA, June 1986. Elsevier North-Holland, Inc. ISBN 9780444700186. URL https://dl.acm.org/doi/10.5555/26036.26046. [p631]

A. Buja, D. Cook, D. Asimov, and C. Hurley. Computational methods for high-dimensional rotations in data visualization. *Handbook of Statistics*, 24:391–413, 2005. URL https://doi.org/10.1016/S0169-7161(04)24014-7. [p624, 625, 626]

A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*. SIAM, 2009. URL https://doi.org/10.1137/1.9780898718768. [p625]

D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997. URL https://doi.org/10.2307/1390747. [p626]

D. Cook, A. Buja, and J. Cabrera. Projection pursuit indexes based on orthonormal function expansions. *Journal of Computational and Graphical Statistics*, 2(3):225–250, 1993. URL https://doi.org/10.2307/1390644. [p625]

D. Cook, A. Buja, J. Cabrera, and C. Hurley. Grand tour and projection pursuit. *Journal of Computational and Graphical Statistics*, 4(3):155–172, 1995. URL https://doi.org/10.1080/10618600.1995.10474674. [p625]

D. Cook, A. Buja, E.-K. Lee, and H. Wickham. Grand tours, projection pursuit guided tours, and manual controls. In *Handbook of Data Visualization*, pages 295–314. Springer, 2008. URL https://doi.org/10.1007/978-3-540-33037-0_13. [p624, 626, 632]

P. Diaconis and D. Freedman. Asymptotics of graphical projection pursuit. *The Annals of Statistics*, pages 793–815, 1984. URL https://doi.org/10.1214/aos/1176346703. [p625]

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 100(9):881–890, 1974. URL https://doi.org/10.1109/T-C.1974.224051. [p625]

V. Granville, M. Krivánek, and J.-P. Rasson. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994. URL https://doi.org/10.1109/34.295910. [p625]

P. Hall. On polynomial-based projection indices for exploratory projection pursuit. *The Annals of Statistics*, 17(2):589–605, 1989. URL https://doi.org/10.1214/aos/1176347127. [p625]

K. Healy. *Data visualization: a practical introduction*. Princeton University Press, 2018. URL https://socviz.co/. [p624]

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998. URL https://doi.org/10.1023/A:1008306431147. [p624]

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. URL https://doi.org/10.1126/science.220.4598.671. [p625]

U. Laa and D. Cook. Using tours to visually investigate properties of new projection pursuit indexes with application to problems in physics. *Computational Statistics*, pages 1–35, 2020. URL https://doi.org/10.1007/s00180-020-00954-8. [p626]

E. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics*, 14(4):831–846, 2005. URL https://doi.org/10.1198/106186005X77702. [p625]

E.-K. Lee and D. Cook. A projection pursuit index for large p small n data. *Statistics and Computing*, 20(3):381–392, 2010. URL https://doi.org/10.1007/s11222-009-9131-1. [p625]

M. Li, Z. Zhao, and C. Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 2020. URL https://doi.org/10.23915/distill.00025. [p624]

N. Loperfido. Skewness-based projection pursuit: A computational approach. *Computational Statistics and Data Analysis*, 120(C):42–57, 2018. URL https://doi.org/10.1016/j.csda.2017.11.001. [p626]

N. Loperfido. Kurtosis-based projection pursuit for outlier detection in financial time series. *The European Journal of Finance*, 26(2-3):142–164, 2020. URL https://doi.org/10.1080/1351847X.2019.1647864. [p626]

D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. *Advances in Applied Probability*, 18(3):747–771, 1986. URL https://doi.org/10.1109/CDC.1985.268600. [p625]

C. Posse. Projection pursuit exploratory data analysis. *Computational Statistics & Data Analysis*, 20(6): 669–687, 1995. URL https://doi.org/10.1016/0167-9473(95)00002-8. [p625, 627]

L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013. URL https://doi.org/10.1007/s10898-012-9951-y. [p625]

H. E. Romeijn. *Random search methods*, pages 3245–3251. Springer US, Boston, MA, 2009. URL https://doi.org/10.1007/978-0-387-74759-0_556. [p625]

B. Schloerke. *geozoo: Zoo of Geometric Objects*, 2016. URL https://CRAN.R-project.org/package=geozoo. R package version 0.5.1. [p630, 631]

J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005. URL https://www.jhuapl.edu/ISSO/. [p625]

J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, MA, 1977. [p624]

A. Unwin. *Graphical data analysis with R*, volume 27. CRC Press, 2015. URL https://doi.org/10.1201/9781315370088. [p624]

R. C. White. A survey of random methods for parameter optimization. *Simulation*, 17(5):197–205, Nov. 1971. URL https://doi.org/10.1177/003754977101700504. [p625]

H. Wickham. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. URL https://doi.org/10.18637/jss.v059.i10. [p628]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL https://doi.org/10.1007/978-0-387-98141-3. [p628]

H. Wickham, D. Cook, H. Hofmann, and A. Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. URL http://doi.org/10.18637/jss.v040.i02. [p626, 627, 636]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2020. URL https://CRAN.R-project.org/package=dplyr. R package version 1.0.2. [p628]

C. O. Wilke. *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019. URL https://clauswilke.com/dataviz/. [p624]

L. Wilkinson and G. Wills. Scagnostics distributions. *Journal of Computational and Graphical Statistics*, 17 (2):473–491, 2008. [p626]

L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 157–164, Oct 2005. [p626]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL https://yihui.name/knitr/. ISBN 978-1498716963. [p638]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. ISBN 978-1138359338. [p638]

Z. B. Zabinsky. *Stochastic adaptive search for global optimization*, volume 72. Springer Science & Business Media, 2013. URL https://doi.org/10.1007/978-1-4419-9182-9. [p625]

H. S. Zhang, D. Cook, U. Laa, N. Langrené, and P. Menéndez. *ferrn: Facilitate Exploration of touRR optimisatioN*, 2021. URL https://github.com/huizezhang-sherry/ferrn/. R package version 0.0.1. [p624, 636]

H. Sherry Zhang
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
huize.zhang@monash.edu

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
dicook@monash.edu

Ursula Laa
University of Natural Resources and Life Sciences
Institute of Statistics
Vienna, Austria
ursula.laa@boku.ac.at

Nicolas Langrené
CSIRO Data61
34 Village Street, Docklands VIC 3008 Australia
Melbourne, Australia
nicolas.langrene@csiro.au

Patricia Menéndez
Monash University
Department of Econometrics and Business Statistics
Melbourne, Australia
patricia.menendez@monash.edu

volesti: Volume Approximation and Sampling for Convex Polytopes in R

by Apostolos Chalkis and Vissarion Fisikopoulos

Abstract Sampling from high-dimensional distributions and volume approximation of convex bodies are fundamental operations that appear in optimization, finance, engineering, artificial intelligence, and machine learning. In this paper, we present **volesti**, an R package that provides efficient, scalable algorithms for volume estimation, uniform, and Gaussian sampling from convex polytopes. **volesti** scales to hundreds of dimensions, handles efficiently three different types of polyhedra and provides non existing sampling routines to R. We demonstrate the power of **volesti** by solving several challenging problems using the R language.

Introduction

High-dimensional sampling from multivariate distributions with Markov Chain Monte Carlo (MCMC) algorithms is a fundamental problem with many applications in science and engineering (Iyengar, 1988; Somerville, 1998; Genz and Bretz, 2009; Schellenberger and Palsson, 2009; Venzke et al., 2021). In particular, multivariate integration over a convex set and volume approximation of such sets —a special case of integration— have accumulated a broad amount of effort over the last decades. Nevertheless, those problems are computationally hard for general dimensions (Dyer and Frieze, 1988). MCMC algorithms have made remarkable progress efficiently solving the problems of sampling and volume estimation of convex bodies while enjoying great theoretical guarantees (Chen et al., 2018; Lee and Vempala, 2018; Mangoubi and Vishnoi, 2019). However, theoretical algorithms cannot be applied efficiently to real-life computations. For example, the asymptotic analysis by Lovász and Vempala (2006) hides some large constants in the complexity, and in Lee and Vempala (2018), the step of the random walk used for sampling is too small to be an efficient choice in practice. Therefore, practical algorithms have been designed by relaxing the theoretical guarantees and applying new algorithmic and statistical techniques to perform efficiently while at the same time meeting the requirements for high accuracy results (Emiris and Fisikopoulos, 2014; Cousins and Vempala, 2016; Chalkis et al., 2019).

In this paper, we present **volesti** (Fisikopoulos et al., 2020), an R package containing a variety of high-dimensional MCMC methods for sampling from multivariate distributions restricted to a convex polytope and randomized algorithms for volume estimation of convex polytopes. In particular, it includes efficient implementations of three practical volume algorithms—Sequence of Balls (SoB) (Emiris and Fisikopoulos, 2014), Cooling Gaussians (CG) (Cousins and Vempala, 2016), and Cooling convex Bodies (CB) (Chalkis et al., 2019). In addition to volume estimation, **volesti** provides efficient implementations for Random-Directions and Coordinate-Directions Hit and Run (RDHR and CDHR) (Smith, 1984), Ball Walk (BaW) (Hastings, 1970), Billiard Walk (BiW) (Polyak and Gryazina, 2014). The first three can be used to sample from multivariate uniform or spherical Gaussian distributions (centered at any point), while BiW can be employed, by definition, only for uniform sampling. On the whole, **volesti** is the first R package that:

(a) performs high-dimensional volume estimation,

(b) efficiently handles three different types of polyhedra in high dimensions, namely H-polytopes, V-polytopes, and Z-polytopes,

(c) provides—previously absent from R—MCMC sampling algorithms for uniform and truncated Gaussian distributions, namely BaW, CDHR, and BiW,

(d) solves some challenging problems in finance, engineering, and applied mathematics.

On top of **volesti** presentation, we illustrate the usage of **volesti** in the study of convergence of various random walks (e.g., Figure 3) and accuracy of volume estimation methods. Regarding applications, in the last section, we illustrate how one can (a) exploit **volesti** to detect shock events in stock markets following the results by Calès et al. (2018), (b) evaluate zonotope approximation in engineering (Kopetzki et al., 2017), and (c) approximate the number of linear extensions of a partially ordered set, which is useful in various applications in artificial intelligence and machine learning.

To improve the presentation of the current paper, detailed comparisons and benchmarking of R packages–including **volesti**–for solving the problems of MCMC sampling, volume computation, and numerical integration are presented in a separate blog post (Chalkis and Fisikopoulos, 2021).

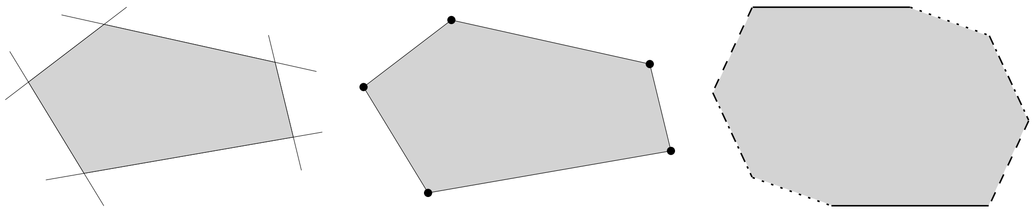


Figure 1: Examples of three different polytope representations. From left to right: an H-polytope, a V-polytope, and a Z-polytope (a sum of four segments).

Related R software and applications

Considering MCMC methods to sample from multivariate distributions are divided into two main categories: truncated to a convex body and untruncated distributions. For the first category—which clearly is the main focus of this paper—an important case is the truncated Gaussian distribution which arises in several applications in statistics. Bolin and Lindgren (2015) sample from truncated Gaussian distributions in a novel importance sampling method to study Markov processes that exceed a certain level. Wadsworth and Tawn (2014) use sampling from a specific truncated Gaussian distribution to develop a novel method for likelihood inference, while Huser and Davison (2013) sample from the same distribution for likelihood estimation for max-stable processes. In curve prediction, they exploit Gaussian sampling to compute simultaneous confidence bands to forecast a full curve from explanatory variables (Azaïs et al., 2010). Grün and Hornik (2012) study the posterior distribution for Bayesian inference on mixed regression models to represent human immunodeficiency virus ribonucleic acid levels, a Gaussian restricted to a convex polytope. In Albert and Chib (1993), the probit regression model for binary outcomes have an underlying normal regression structure on latent continuous data; sampling from the posterior distribution of the parameters involves sampling from a truncated Gaussian distribution.

Another important special case is the truncated uniform distribution. In systems biology the flux space of a metabolic network is represented by a convex polytope (Haraldsdòttir et al., 2017); uniform sampling from the interior of that polytope could lead to important biological insights. In computational finance, the set of all possible portfolios in a stock market is in general a convex polytope. Volume computation and uniform sampling from that set is useful for crises detection (Calès et al., 2018) and efficient portfolio allocation and analysis (Pouchkarev et al., 2004; Hallerbach et al., 2002).

Considering R packages for the truncated case, there is **tmg** (Pakman, 2015) implementing exact Hamiltonian Monte Carlo (HMC) with boundary reflections as well as **multinomineq** (Heck, 2019), **lineqGPR** (Lopez, 2019), **restrictedMVN** (Taylor and Benjamini, 2016), **tmvmixnorm** (Ma et al., 2020) implementing variations of the Gibbs sampler. To our knowledge, the only two R packages for uniform sampling is **hitandrun** (van Valkenhoef and Tervonen, 2019) and **limSolve** (den Meersche K. et al., 2009), which exposes the R function `xsample()` (den Meersche et al., 2009). For the untruncated case, packages **HybridMC** (Morey, 2009), **rhmc** (Sartório, 2018), **mcmc** (Geyer and Johnson, 2020), and **MHadaptive** (Chivers, 2012) provide implementations for HMC and Metropolis Hastings algorithms, respectively. For volume computation, the only existing package, **geometry** (Roussel et al., 2019), computes the volume of the convex hull of a set of points and is based on the C++ library, qhull (Barber et al., 1996).

Algorithms and polytopes

Convex polytopes

Convex polytopes are a special case of convex bodies with special interest in many scientific fields and applications. For example, in optimization, the feasible region of a linear program is a polytope, and in finance, the set of portfolios is usually expressed by a polytope (i.e., the simplex). More formally, an H-polytope is defined as

$$P := \{x \mid Ax \leq b\} \subseteq \mathbb{R}^d,$$

where $A \in \mathbb{R}^{m\times d}$ and $b \in \mathbb{R}^m$, and we say that P is given in H-representation. Each row $a_i^T \in \mathbb{R}^d$ of matrix A corresponds to a normal vector that defines a halfspace $a_i^T x \leq b_i,\ i = [m]$. The intersection of those halfspaces defines the polytope P, and the hyperplanes $a_i^T x = b_i,\ i = [m]$ are called facets of P. A V-polytope is given by a matrix $V \in \mathbb{R}^{d\times n}$, which contains n points column-wise, and we say that P is given in V-representation. The points of P that cannot be written as convex combinations of

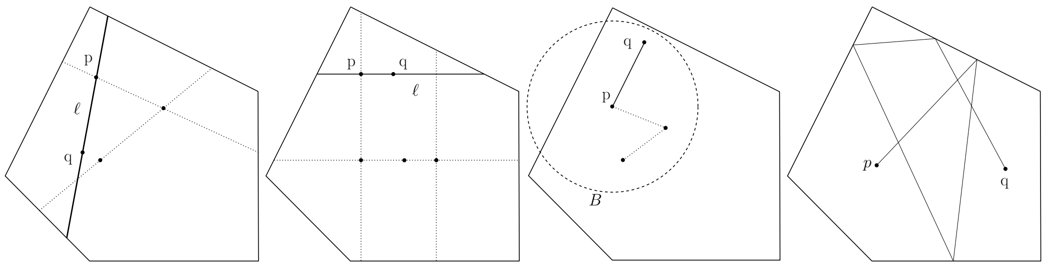


Figure 2: Examples of random walks. From left to right: RDHR, CDHR, BaW, BiW; p is the point at current step and q the new point computed; ℓ is the line computed by RDHR and CDHR; B is the ball computed by BaW. Dotted lines depict previous steps.

other points of P are called vertices. The polytope P is defined as the convex hull of those vertices, i.e., the smallest convex set that contains them. Equivalently, a V-polytope can be seen as the linear map of the canonical simplex $\Delta^{n-1} := \{x \in \mathbb{R}^n \mid x_i \geq 0,\ \sum_{i=1}^{n} x_i = 1\}$ according to matrix V, i.e.,

$$P := \{x \in \mathbb{R}^d \mid \exists y \in \Delta^{n-1} : x = Vy\}$$

A Z-polytope (or zonotope) is given by a matrix $G \in \mathbb{R}^{d\times k}$, which contains k segments column-wise, which are called generators. In this case, P is defined as the Minkowski sum of those segments and we say that it is given in Z-representation. We call *order* of a Z-polytope the ratio between the number of segments over the dimension. Equivalently, P can be expressed as the linear map of the hypercube $[-1,1]^k$ with matrix G, i.e.

$$P := \{x \in \mathbb{R}^d \mid \exists y \in [-1,1]^k : x = Gy\}.$$

Thus, a Z-polytope is a centrally symmetric convex body, as a linear map of an other centrally symmetric convex body. Examples of an H-polytope, a V-polytope and a Z-polytope in two dimensions are illustrated in Figure 1. For an excellent introduction to polytope theory, we recommend the book of Ziegler (1995).

MCMC sampling and geometric random walks

We more formally define here the four geometric random walks implemented in **volesti**, namely, Hit and Run (two variations, RDHR, and CDHR), Ball walk (Baw), and Billiard walk (BiW). They are illustrated in Figure 2 for two dimensions.

In general, if $f : \mathbb{R}^n \to \mathbb{R}_+$ is a non-negative integrable function, then it defines a measure π_f on any measurable subset A of $\mathbb{R}^d$,

$$\pi_f(A) = \frac{\int_A f(x)dx}{\int_{\mathbb{R}^d} f(x)dx}$$

Let ℓ be a line in $\mathbb{R}^d$ and let $\pi_{\ell,f}$ be the restriction of π to ℓ,

$$\pi_{\ell,f}(P) = \frac{\int_{p+tu\in P} f(p+tu)dt}{\int_\ell f(x)dx},$$

where p is a point on ℓ and u a unit vector parallel to ℓ.

Algorithm 1 describes the general Hit and Run procedure. When the line ℓ in line (1.) of the pseudocode is chosen uniformly at random from all possible lines passing through p, then the walk is called Random-Directions Hit and Run (Smith, 1984). To pick a random direction through point $p \in \mathbb{R}^d$, we could sample d numbers $g_1, \ldots, g_d$ from $\mathcal{N}(0,1)$, and then the vector $u = (g_1, \ldots, g_d)/\sqrt{\sum g_i^2}$ is uniformly distributed on the surface of the d-dimensional unit ball. A special case is called Coordinate-Directions Hit and Run (Smith, 1984), where we pick ℓ uniformly at random from the set of d lines that passing through p and are parallel to the coordinate axes.

```
Algorithm 1: Hit_and_run($P, p, f$)
  Input  : Polytope $P \subset \mathbb{R}^d$, point $p \in P$, $f : \mathbb{R}^d \to \mathbb{R}_+$
  Output: A point $q \in P$
    1. Pick a line $\ell$ through $p$.
    2. return a random point on the chord $\ell \cap P$ chosen from the distribution $\pi_{\ell,f}$.
```

The Ball walk (Algorithm 2) needs, additionally to Hit and Run, a radius δ as input. In particular, Ball walk is a special case of Metropolis Hastings (Hastings, 1970) when the target distribution is truncated. For both Hit and Run and Ball walk π_f, is the stationary distribution of the random walk. If $f(x) = e^{-||x-x_0||^2/2\sigma^2}$, then the target distribution is the multidimensional spherical Gaussian with variance σ^2 and its mode at x_0. When f is the indicator function of P, then the target distribution is the uniform distribution.

```
Algorithm 2: Ball_walk($P, p, \delta, f$)
  Input  : Polytope $P \subset \mathbb{R}^d$, point $p \in P$, radius $\delta$, $f\mathbb{R}^d \to \mathbb{R}_+$
  Output: A point $q \in P$
    1. Pick a uniform random point $x$ from the ball of radius $\delta$ centered at $p$
    2. return $x$ with probability $\min\left\{1, \frac{f(x)}{f(p)}\right\}$; return $p$ with the remaining probability.
```

Billiard walk is a random walk for sampling from the uniform distribution (Polyak and Gryazina, 2014). It tries to emulate the movement of a gas particle during the physical phenomena of filling uniformly a vessel. Algorithm 3 implements Billiard walk, where $\langle \cdot, \cdot \rangle$ is the inner product between two vectors, $|| \cdot ||$ is the ℓ^2 norm, and $| \cdot |$ is the length of a segment.

```
Algorithm 3: Billiard_walk($P, p, \tau, R$)
  Input  : Polytope $P \subset \mathbb{R}^d$, current point of the random walk $p \in P$, length of trajectory
           parameter $\tau \in \mathbb{R}_+$, upper bound on the number of reflections $R \in \mathbb{N}$
  Output: A point $q \in P$
    1. Set the length of the trajectory $L \leftarrow -\tau \ln \eta$, $\eta \sim \mathcal{U}(0,1)$;
       Set the number of reflections $n \leftarrow 0$ and $p_0 \leftarrow p$;
       Pick a uniformly distributed direction on the unit sphere, $v$;
    2. Update $n \leftarrow n + 1$; If $n > R$ return $p_0$;
    3. Set $\ell \leftarrow \{p + tv, 0 \leq t \leq L\}$;
    4. If $\partial P \cap \ell = \emptyset$ return $p + Lv$;
    5. Update $p \leftarrow \partial P \cap \ell$; Let $s$ be the inner normal vector of the tangent plane on $p$, s.t. $||s|| = 1$;
       Update $L \leftarrow L - |P \cap \ell|$, $v \leftarrow v - 2\langle v, s\rangle s$; goto 2;
```

Every random walk starts from a point in the convex body and perform a number of steps called *walk length*. The larger the walk length is, the less correlated the final with the starting point will be. The number of steps to get an uncorrelated point, that is, a point approximately drawn from π_f is called *mixing time*. The number of operations performed to generate a point is called *cost per step*. Hence, the total cost to generate a random point is the mixing time multiplied by the cost per step.

random walk	mixing time	cost/step H-polytope	cost/step V- & Z-polytope
RDHR (Lovász and Vempala, 2006)	$O^*(d^3)$	$O(md)$	2 LPs
CDHR (Laddha and Vempala, 2020)	$O^*(d^{10})$	$O(m)$	2 LPs
BaW (Lee and Vempala, 2017)	$O^*(d^{2.5})$	$O(md)$	1 LP
BiW (Polyak and Gryazina, 2014)	**?**	$O((d+R)m)$	R LPs

Table 1: Overview of the random walks implemented in **volesti**. LP for linear program; R for the number of reflections per point in BiW; D for the diameter of the polytope.

Table 1 displays known complexities for mixing time and cost per step. For the mixing time of RDHR, we assume that P is well rounded, i.e., $B_d \subseteq P \subseteq C\sqrt{d}B_d$, where B_d is the unit ball and C a constant. In general, if $rB_d \subseteq P \subseteq RB_d$ then RDHR mixing time is $O^*(d^2(R/r)^2)$. For the mixing time of Ball walk in Table 1, we assume that P is in isotropic position and the radius of the ball is $\delta = \Theta(1/\sqrt{d})$ (Lee and Vempala, 2017). There are no theoretical bounds on mixing time for CDHR and BiW. Polyak and Gryazina (2014) experimentally shows that BiW converges faster than RDHR when $\tau \approx diam(P)$, i.e., the diameter of P. CDHR is the main paradigm for sampling in practice from H-polytopes, e.g., in volume computation (Emiris and Fisikopoulos, 2014) and biology (Haraldsdòttir et al., 2017). The main reason behind this is the small cost per step and the same convergence in practice as RDHR (Emiris and Fisikopoulos, 2014). For V- and Z-polytopes, the cost per step of BiW is comparable with that of CDHR. Moreover, it converges fast to the uniform distribution (Chalkis et al., 2019). The fact that all above walks are implemented in **volesti** enable us to empirically evaluate their mixing time using R (e.g., Figure 3).

Volume estimation

As mentioned before, volume computation is a hard problem, so given a polytope P, we have to employ randomized algorithms to approximate $\text{vol}(P)$ within some target relative error ϵ and high probability. The keys to the success of those algorithms are the Multiphase Monte Carlo (MMC) technique and sampling from multivariate distributions with geometric random walks.

In particular, we define a sequence of functions $\{f_0, \dots, f_q\}$, $f_i : \mathbb{R}^d \to \mathbb{R}$. Then, $\text{vol}(P)$ is given by the following telescopic product:

$$\text{vol}(P) = \int_P dx = \int_P f_q(x)dx \frac{\int_P f_{q-1}(x)dx}{\int_P f_q(x)dx} \cdots \frac{\int_P dx}{\int_P f_0(x)dx} \quad (1)$$

Then, we need to:

- Fix the sequence such that q is as small as possible.
- Select f_i such that each integral ratio can be efficiently estimated.
- Estimate $\int_P f_q(x)dx$.

For a long time researchers, e.g., Lovász et al. (1997), set f_i to be indicator functions of concentric balls intersecting P. It follows that $\int_P f_i(x)dx = \text{vol}(B_i \cap P)$, and the sequence of convex bodies $P = P_1 \supseteq \cdots \supseteq P_q$, $P_i = B_i \cap P$ forms a telescopic product of ratios of volumes, while for $\text{vol}(P_q)$ there is a closed formula. Assuming $rB_d \subset P \subset RB_d$, then $q = O(d \lg R/r)$. The trick now is that we do not have to compute the exact value of each ratio $r_i = \text{vol}(P_i)/\text{vol}(P_{i+1})$, but we can use sampling-rejection to estimate it within some target relative error ϵ_i. If r_i is bounded, then $O(1/\epsilon_i^2)$ uniformly distributed points in P_{i+1} suffices. Another crucial aspect is the sandwiching ratio R/r of P which has to be as small as possible. This was tackled by a rounding algorithm, that is bringing P to nearly isotropic position (Lovász et al., 1997).

The SoB algorithm follows this paradigm and deterministically defines the sequence of P_i such that $0.5 \leq \text{vol}(P_i)/\text{vol}(P_{i+1}) \leq 1$. In the CG algorithm, each f_i is a spherical multidimensional Gaussian distribution, and the algorithm uses an annealing schedule (Lovász and Vempala, 2006) to fix the sequence of those Gaussians. The SoB algorithm uses a similar annealing schedule but to fix a sequence of convex bodies P_i. As far as performance is concerned, the CB algorithm is the most efficient choice for H-polytopes in less than 200 dimensions and for V- and Z-polytopes in any dimension. For the rest of the cases, the user should choose CG algorithm.

Package

The package **volesti** combines the efficiency of C++ and the popularity and usability of R. The package uses the eigen library (Guennebaud et al., 2010) for linear algebra, lpsolve library (Berkelaar et al., 2004) for solving linear programs, and boost random library (Maurer and Watanabe, 2017) (part of Boost C++ libraries) for random numbers and distributions. All the code development is performed on github platform. The package is available in Comprehensive R Archive Network (CRAN) and is regularly updated with new features and bug fixes. We employ continuous integration to test the package on various systems and deploy environments. There is detailed documentation of all the exposed R classes and functions publicly available. We maintain a contribution tutorial to help users and researchers who want to contribute to the development or propose a bug-fix. The package is shipped under the `LGPL-3` license to be open to all the scientific communities. We use **Rcpp** (Eddelbuettel et al., 2020b) to interface C++ with R. In particular, we create one **Rcpp** function for each procedure (such as sampling, volume estimation etc.) and we export it as an R function.

In the following sections, we demonstrate the use of **volesti**. The R scripts in the following sections use only standard R functions, **volesti**. In a single script in Section 2.4, we use **Rfast** to compute the assets' compound return in a stock market.

Polytope classes and generators

The package **volesti** comes with three classes to handle different representations of polytopes. Table 2 demonstrates the exposed R classes. The names of the classes are the names of polytope representations as defined in the previous section. Each polytope class has a few variable members that describe a specific polytope, demonstrated in Table 2. The matrices and the vectors in Table 2 correspond to those in the polytope definitions. The `integer` variable `type` implies the representation: `1` is for H-polytopes, `2` for V-polytopes, `3` for Z-polytopes. The `numerical` variable `volume` corresponds to the volume of the polytopes if it is known. **volesti** provides standard and random polytope generators. The first produce well-known polytopes such as cubes, cross polytopes, and simplices and assign the value of the exact volume to `volume` variable. The second are random generators using various probability distributions and methods to produce a variety of different random polytopes; notably the generated polytopes have unknown volume.

Class	Constructor	Variable members
`"Hpolytope"`	`Hpolytope(A,b)`	$A \in \mathbb{R}^{m\times d}, b \in \mathbb{R}^m$, `integer type`, `numerical volume`
`"Vpolytope"`	`Vpolytope(V)`	$V \in \mathbb{R}^{n\times d}$, `integer type`, `numerical volume`
`"Zonotope"`	`Zonotope(G)`	$G \in \mathbb{R}^{k\times d}$, `integer type`, `numerical volume`

Table 2: Overview of the polytopes' classes in **volesti**.

Uniform sampling from polytopes

A core feature of **volesti** is approximate sampling from convex bodies with uniform or spherical Gaussian target distribution using the four geometric random walks defined above.

The following R script samples 1000 points from the 100-dimensional hypercube $[-1,1]^{100}$ defined as P and stores them in a list.

```
R> d = 100
R> P = gen_cube(d, 'H')

R> samples = sample_points(P, random_walk = list(
                          "walk" = "RDHR", "burn-in"=1000, "walk_length" = 5),
                          n = 1000)
```

We use the Random Directions Hit-and-Run (RDHR) walk. Other choices are: Coordinate Directions Hit-and-Run (CDHR), Ball Walk (BaW), and Billiard Walk (BiW). Setting the parameter `burn-in` to 1000 means that **volesti** burns the first 1000 points RDHR generates; setting walk_length to 5 means that we keep in the list, one every five generated points. The default choice for the target distribution is the uniform distribution.

To evaluate the efficiency of **volesti** sampling routines, one could measure the run-time and estimate the effective sample size (Geyer, 2011) per second. To estimate the effective sample size in R, a standard choice is the package **coda** (Plummer et al., 2020). In Chalkis and Fisikopoulos (2021), benchmarks show that **volesti** can be up to $\sim 2\,500$ times faster than **hitandrun** for uniform sampling from a polytope.

Moreover, using **volesti** and R, we can empirically study the mixing time of the geometric random walks implemented in **volesti**. To this end, we uniformly sample from a random rotation of the 200-dimensional hypercube $[-1,1]^{200}$. First, we generate the hypercube and use `rotate_polytope()` that returns the rotated polytope and the matrix of the linear transformation.

```
R> d = 200
R> num_of_points = 1000
R> P = gen_cube(d, 'H')
R> retList = rotate_polytope(P, rotation = list("seed" = 5))
R> T = retList$T
R> P = retList$P
```

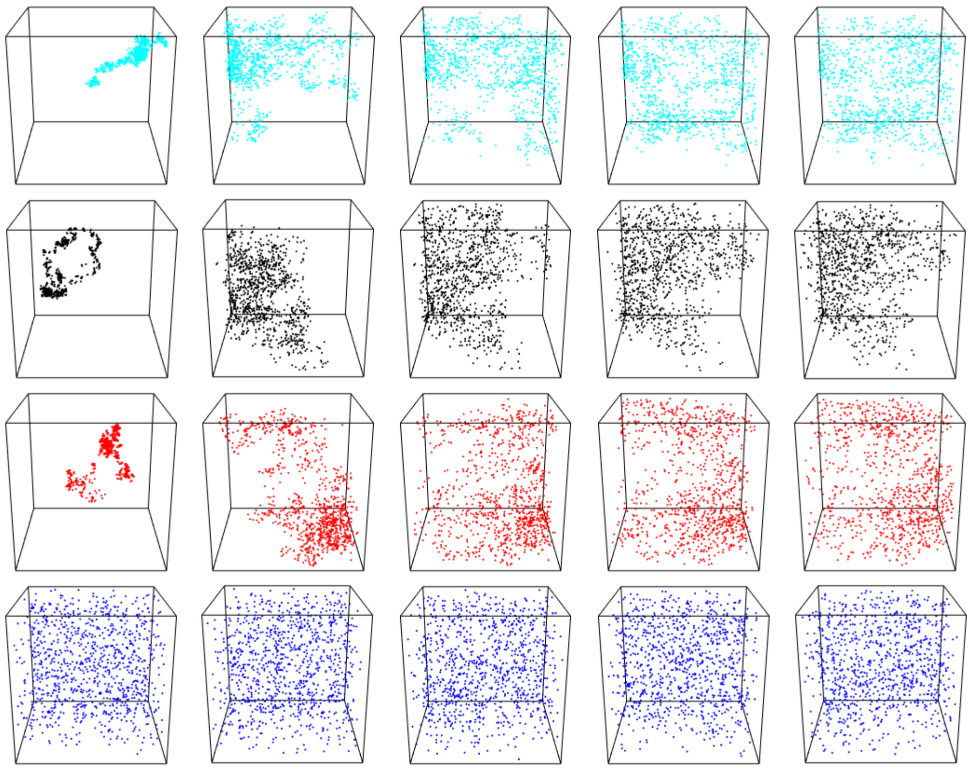


Figure 3: Uniform sampling from a random rotation of the hypercube $[-1,1]^{200}$. We map the sample back to $[-1,1]^{200}$, and then we project them on $\mathbb{R}^3$ by keeping the first three coordinates. Each row corresponds to a different walk: BaW, CDHR, RDHR, BiW. Each column to a different walk length: {1, 50, 100, 150, 200}. That is, the sub-figure in the third row and the forth column corresponds to RDHR with 150 walk length.

Then, we use `sample_points()` to sample from the rotated cube with various walk lengths to test the practical mixing of the random walk.

```
R> for (i in c(1, seq(from = 50, to = 200, by = 50))){
     points1 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                  "walk" = "BaW", "walk_length" = i, "seed" = 5))
     points2 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                  "walk" = "CDHR", "walk_length" = i, "seed" = 5))
     points3 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                  "walk" = "RDHR", "walk_length" = i, "seed" = 5))
     points4 = t(T) %*% sample_points(P, n = num_of_points, random_walk = list(
                                  "walk" = "BiW", "walk_length" = i, "seed" = 5))
   }
```

Finally, we map the points back to $[-1,1]^{200}$ using the inverse transformation, and then we project all the sample points on $\mathbb{R}^3$, or equivalently on the 3D cube $[-1,1]^3$, by keeping the first three coordinates. We plot the results in Figure 3.

Note that, in general, perfect uniform sampling in the rotated polytope would result to perfect uniformly distributed points in the 3D cube $[-1,1]^3$. Hence, Figure 3 shows an advantage of BiW in mixing time for this scenario compared to the other walks—it mixes relatively well even with one step (i.e. walk length). Notice also that the mixing of both CDHR and RDHR seem similar while it is slightly better than the mixing of BaW.

Gaussian sampling from polytopes

In many Bayesian models, the posterior distribution is a multivariate Gaussian distribution restricted to a specific domain. We illustrate the usage of **volesti** for the case of the truncation being the canonical simplex $\Delta^n = \{x \in R^n \mid x_i \geq 0, \sum_i x_i = 1\}$, which is of special interest. This situation typically occurs whenever the unknown parameters can be interpreted as fractions or probabilities. Thus, it appears in many important applications (Altmann et al., 2014). In particular, we consider the following density,

$$f(x|\mu,\Sigma) \propto \begin{cases} exp[-\frac{1}{2}(x-\mu)^T\Sigma(x-\mu)], & \text{if } x \in \Delta^n, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, the support of the density in Equation (2) is defined by a convex subset of a linear subspace of $\mathbb{R}^n$. Thus, to sample from $f(x|\mu,\Sigma)$, we apply a proper linear transformation, induced by a matrix $N \in \mathbb{R}^{n\times(n-1)}$ that maps the support to a full-dimensional polytope in $\mathbb{R}^{n-1}$, while the covariance matrix changes accordingly to $\Sigma' = N^T\Sigma N$. Then, we apply a Cholesky decomposition to $\Sigma' = LL^T$ and employ the linear transformation induced by L to transform the distribution into a spherical Gaussian distribution.

In the following R script we first generate a random 100-dimensional positive definite matrix Σ. Then, we sample from the multivariate Gaussian distribution with the covariance matrix being Σ and the mode being the center of the canonical simplex Δ^n. To achieve this goal, we first apply all the necessary linear transformations to both the probability density function and the Δ^n to obtain the standard Gaussian distribution, $\mathcal{N}(0, I_n)$, restricted to a general full-dimensional simplex.

```
R> d = 100
R> S = matrix( rnorm(d*d,mean=0,sd=1), d, d) #random covariance matrix
R> S = S %*% t(S)
R> shift = rep(1/d, d)
R> A = -diag(d)
R> b = rep(0,d)
R> b = b - A %*% shift
R> Aeq = t(as.matrix(rep(1,d), 10,1))
R> N = pracma::nullspace(Aeq)
R> A = A %*% N #transform the truncation into a full dimensional polytope
R> S = t(N) %*% S %*% N
R> A = A %*% t(chol(S)) #Cholesky decomposition to transform to the standard Gaussian
R> P = Hpolytope(A=A, b=as.numeric(b)) #new truncation
```

Next, we use the `sample_points()` function to sample from the standard Gaussian distribution restricted to the computed simplex, and we apply the inverse transformations to obtain a sample in the initial space.

```
R> samples = sample_points(P, n = 100000, random_walk =
                           list("walk"="CDHR", "burn-in"=1000,
                           "starting_point" = rep(0, d-1),
                           distribution = list("density" = "gaussian",
                           "mode" = rep(0, d-1))))
R> samples_initial_space = N %*% samples +
   kronecker(matrix(1, 1, 100000), matrix(shift, ncol = 1))
```

In the previous script, we set the starting point of the walk to the mode of the Gaussian, i.e., the origin. Note that the default choice in **volesti** for the target distribution in the case of Gaussian sampling is the standard Gaussian; that is, the target distribution in the above script.

Considering comparisons, **volesti** is at least one order of magnitude faster than **restrictedMVN** and **tmg** for computing a sample of similar quality. For more details on comparison with other packages, we refer to (Chalkis and Fisikopoulos, 2021).

Volume estimation

Let us now give an example of how we approximate the volume of a polytope in **volesti**. Since this is a randomized algorithm, it makes sense to compute some statistics for the output values using R when approximating the volume of the 10-dimensional cube $[-1, 1]^{10}$ generated as an H-polytope.

```
R> P = gen_cube(10, 'H')
R> volumes = list()
R> for (i in seq_len(20)) {
     volumes[[i]] = volume(P, settings = list("error" = 0.2))
   }
```

By changing the error to 0.02, we can obtain more accurate results. The results are illustrated in Figure 4. Note that the exact volume is 1024.

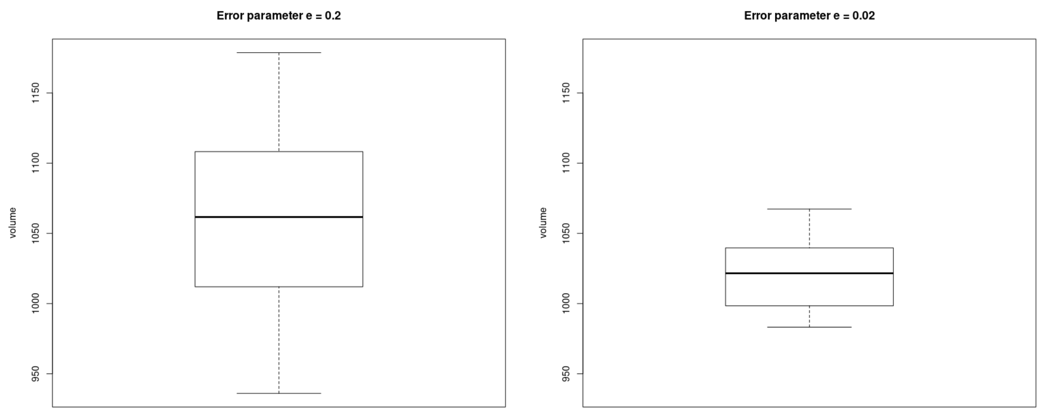


Figure 4: The boxplot of the estimated volumes of the hypercude $[-1,1]^{10}$ by **volesti**. Left, the input error parameter is $\epsilon = 0.2$, and right is $\epsilon = 0.02$.

To understand the need for randomized computation in high dimensions implemented in **volesti**, we can consider the state-of-the-art volume computation in R today, namely, **geometry**. It implements a deterministic algorithm in which run-time grows exponentially with the dimension. Because of the later property, **geometry** generally, fails to terminate for polytope in dimension $d \geq 20$. See (Chalkis and Fisikopoulos, 2021) for comparison details with **geometry**.

The following script illustrates the usage and efficiency of **volesti** to compute the volume of high-dimensional polytopes. In particular, a V-polytope, namely the cross-polytope, and an H-polytope, namely the hypercube.

```
R> d = 80
R> P = gen_cross(80, 'V')  #generate a cross polytope in V-representation

R> time = system.time({
          volume_estimation = volume(P, settings = list(
                                     "algorithm" = "CB", "random_walk" = "BiW",
                                     "seed" = 127)) })
R> exact_volume = 2^d/prod(1:d)
R> cat(time[1], abs(volume_estimation - exact_volume) / exact_volume)

82.874   0.074434

R> P = gen_cube(d, 'H')   #generate a hypercube polytope in H-representation

R> time = system.time({
          volume_estimation = volume(P, settings = list(
                                     "algorithm" = "CB", "random_walk" = "CDHR",
                                     "seed" = 23)) })
R> exact_volume = 2^d
R> cat(time[1], abs(volume_estimation - exact_volume) / exact_volume)

0.657   0.067633
```

For V- and Z- polytopes the most efficient choice of random walk is BiW, while for H-polytopes is CDHR. This explains why we use different random walks in the previous script. However, notice that the run-time for the H-polytope is two order of magnitude smaller. This happens because the cost per step of a random walk in a V-polytope increases comparing to H-polytopes.

Last but not least, **volesti** provides random polytope generators. The following command estimates the volume of a randomly generated V-polytope that is the convex hull of 40 uniformly generated random points from the 20-dimensional cube.

```
R> P = gen_rand_vpoly(20, 40, generator = list("body" =  "cube", "seed" = 1729))
R> volume_estimation = volume(P)
```

The next call estimates the volume of an H-polytope randomly generated as an intersection of 180 linear halfspaces computed by random tangent hyperplanes on an 60-dimensional hypersphere.

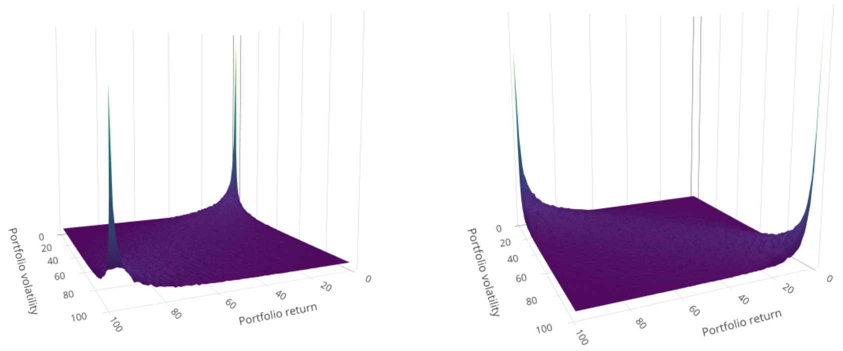


Figure 5: Left, a copula that corresponds to normal period (07/03/2007 − 31/05/2007), $I = 0.2316412$. Right, a copula that corresponds to a crisis period (18/12/2008 − 13/03/2009), $I = 5.610785$; x axis is for return and y axis is for volatility.

```
R> P = gen_rand_hpoly(60, 180, generator = list('constants' = 'sphere'))
R> volume_estimation = volume(P)
```

Since the exact volume of those polytopes is unknown, the accuracy of the computed estimation is unknown and statistical methods such as the effective sample size (Geyer, 2011) could be used.

Applications

We demonstrate **volesti**'s potential to solve challenging problems. More specifically, we provide detailed use-cases for applications in finance (crises detection and portfolio scoring), decision and control, multivariate integration, and artificial intelligence.

Financial crises detection and portfolio scoring

In this subsection, we present how one could employ **volesti** to detect financial crises or shock events in stock markets by following the method of Calès et al. (2018). For all the examples in the sequel, we use a set of 52 popular exchange-traded funds (ETFs) and the US central bank (FED) rate of return publicly available from `https://stanford.edu/class/ee103/portfolio.html`. The following script is used to load the data.

```
R> MatReturns = read.table("https://stanford.edu/class/ee103/data/returns.txt",
                           sep = ",")
R> MatReturns = MatReturns[-c(1, 2), ]
R> dates = as.character(MatReturns$V1)
R> MatReturns = as.matrix(MatReturns[ ,-c(1, 54)])
R> MatReturns = matrix(as.numeric(MatReturns), nrow = dim(MatReturns )[1], ncol =
                       dim(MatReturns )[2], byrow = FALSE)
R> nassets = dim(MatReturns)[2]
```

The method uses the copula representation to capture the dependence between portfolios' returns and volatility. A copula is an approximation of the bivariate joint distribution while both marginals follow the uniform distribution. In normal times, portfolios are characterized by slightly positive returns and moderate volatility, in up-market times (typically bubbles) by high returns and low volatility, and during financial crises by strongly negative returns and high volatility. Thus, when a copula implies a positive dependence (see Figure 5 left), then it probably comes from a normal period. On the other side, when the dependence between portfolios' return and volatility is negative (see Figure 5 right), the copula probably comes from a crisis period. The first case occurs when the indicator that computes the ratio between the red mass over the blue mass is smaller than 1, and the second case when that indicator is larger than 1. The function `copula()` can be used to compute such copulas. When two vectors of returns are given as input by the user, then the computed copula is related to the problem of the momentum effect in stock markets.

The following script produces Figure 5 by setting the starting and the stopping date for the left and the right plot, respectively. To compute the copula, we use the compound asset return, which is the rate of return for capital over a cumulative series of time (Calès et al., 2018).

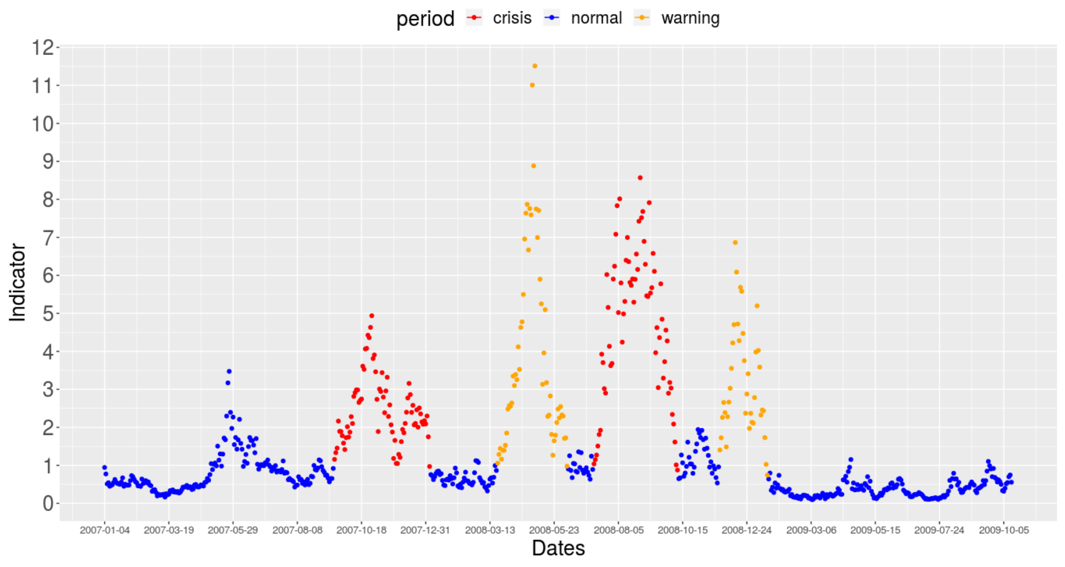


Figure 6: The values of the indicators from 2007-01-04 until 2010-01-04. We mark the crisis periods with red, the warning periods with orange, and the normal periods with blue that **volesti** identifies.

```
R> row1 = which(dates %in% "2008-12-18")
R> row2 = which(dates %in% "2009-03-13")
R> compound_asset_return = Rfast::colprods(1 + MatReturns[row1:row2, ]) - 1
R> mass = copula(r1 = compound_asset_return, sigma = cov(MatReturns[row1:row2, ]),
                 m = 100, n = 1e+06, seed = 5)
```

Moreover, the function `compute_indicators()` computes the copulas of all the sets of `win_len` consecutive days and returns the corresponding indicators and the states of the market during the given time period. The next script takes as input the daily returns of all the 52 assets from 01/04/2007 until 04/01/2010. When the indicator is ≥ 1 for more than 30 days, we issue a warning, and when it is for more than 60 days, we mark this period as a crisis (see Figure 6).

```
R> row1 = which(dates %in% "2007-01-04")
R> row2 = which(dates %in% "2010-01-04")
R> market_analysis = compute_indicators(returns =  MatReturns[row1:row2, ],
                                        parameters = list("win_len" = 60, "m" = 100,
                                        "n" = 1e+06, "nwarning" = 30, "ncrisis" = 60,
                                        "seed" = 5))
R> I = market_analysis$indicators
R> market_states = market_analysis$market_states
```

We compare the results with the database for financial crises in European countries proposed in Duca et al. (2017). The only listed crisis for this period is the sub-prime crisis (from December 2007 to June 2009). Notice that Figure 6 successfully points out 4 crisis events in that period (2 crisis and 2 warning periods) and detects sub-prime crisis as a W-shape crisis.

As a second financial application, we will use **volesti** to evaluate the performance of a given portfolio. In particular, **volesti** computes the proportion of all possible allocations that the given portfolio outperforms. This score independently introduced in Pouchkarev (2005); Guegan et al. (2011); Banerjee and Hung (2011), and is an alternative to more classical choices for the evaluation of the performance of a portfolio as the Sharpe-like ratios proposed in the 1960's by Jensen (1967); Sharpe (1966); Treynor (2015). However, the efficient computation of that score was uncertain until Calès et al. (2018) notice that Varsi's algorithm (Varsi, 1973) can be used to perform robust computations in high dimensions. Varsi's algorithm is implemented in **volesti** by the function `frustum_of_simplex()` and computes volumes in thousands of dimensions in just a few milliseconds on modest hardware. As an example, the following R script let us know that on 03/13/2009, any portfolio with a return of 0.002 outperforms almost 48% of all possible portfolios.

```
R> R = MatReturns[which(dates %in% "2009-03-13"), ]
R> R0 = 0.002
R> tim = system.time({ exact_score = frustum_of_simplex(R, R0) })
R> cat(exact_score, tim[3])
```

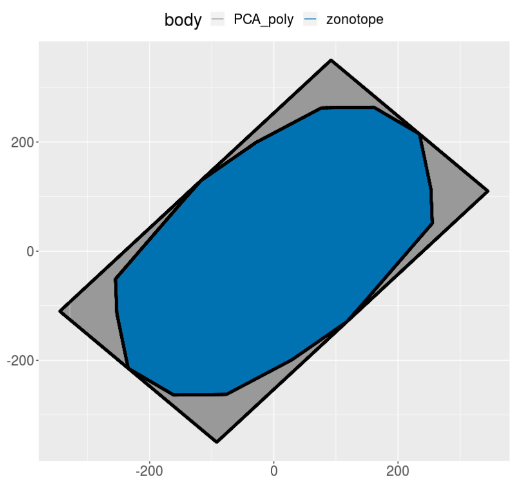


Figure 7: Blue color represents a 2D Z-polytope. Grey color represents the over-approximation of P computed with PCA method.

```
0.4773961  0.001
```

Zonotope volumes in decision and control

Volume approximation for Z-polytopes (or zonotopes) could be very useful in several applications in decision and control (Kopetzki et al., 2017), in autonomous driving (Althoff and Dolan, 2014), or human-robot collaboration (Pereira and Althoff, 2015). The complexity of algorithms that manipulate Z-polytopes strongly depends on their order. Thus, to achieve efficient computations, the common approach in practice is to over-approximate the Z-polytope at hand P, as tight as possible, with a second Z-polytope P_{red} of a smaller order. Then, the ratio of fitness $\rho = (\mathrm{vol}(P_{red})/\mathrm{vol}(P))^{1/d}$ is a good measure for the quality of the approximation. However, this ratio cannot be computed for dimensions typically larger than 10 (see (Kopetzki et al., 2017)). **volesti** is the first software to the best of our knowledge that efficiently approximates the ratio of fitness of a high dimensional Z-polytope–typically up to 100 and order 200–or a Z-polytope of very high order in lower dimensions–e.g., order 1500 in 10 dimensions.

As an illustration, the following R script generates a random 2D zonotope, computes the over-approximation with the PCA method, and estimates the ratio of fitness. The `sample_points` function is then used to plot the two polygons (Figure 7).

```
R> Z = gen_rand_zonotope(2, 8, generator = list("distribution" = "uniform",
                         "seed" = 1729))
R> points1 = sample_points(Z, random_walk = list("walk" = "BRDHR"), n = 10000)
R> retList = zonotope_approximation(Z = Z, fit_ratio = TRUE,
                                     generator = list("seed" = 5))
R> P = retList$P
R> cat(retList$fit_ratio)

1.116799539

R> points2 = sample_points(P, random_walk = list("walk" = "BRDHR", "seed" = 5),
                           n = 10000)
```

High-dimensional integration

Computing the integral of a function over a convex set (i.e., convex polytope) is a hard fundamental problem with numerous applications. **volesti** can be used to approximate the value of such an integral by a simple MCMC integration method, which employs the vol(P) and a uniform sample in P. In particular, let

$$I = \int_P f(x)dx. \tag{3}$$

dimension	*Exact value*	*Estimated value*	*Rel. error*	*Exact Time (sec)*	*Est. Time (sec)*
5	0.02738404	0.02446581	0.1065667	0.023	3.983
10	3.224286e-06	3.204522e-06	0.00612976	3.562	11.95
15	4.504834e-11	4.867341e-11	0.08047068	471.479	33.256
20	-	1.140189e-16	-	-	64.058

Table 3: We compute the integral of the function in Equation (5) over a random generated V-polytope. *Exact value*: the exact value of the integral using **SimplicialCubature** and **geometry**; *Estimated value*: the estimation of the integral with **volesti**; *Rel. error*: the relative error of **volesti**; *Exact Time*: the sum of run-times of **geometry** and **SimplicialCubature**; *Est. Time*: the run-time of **volesti**; "-" indicates that the program halts.

Then, sample N uniformly distributed points $x_1, \dots, x_N$ from P and,

$$I \approx \mathrm{vol}(P) \frac{1}{N} \sum_{i=1}^{N} f(x_i). \tag{4}$$

The following R script generates a V-polytope for $d = 5, 10, 15, 20$, and estimates the integral of

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i + 2x_1^2 + x_2 + x_3, \tag{5}$$

over the generated V-polytope P.

Considering the efficiency of **volesti**, Table 3 reports the exact value of I computed by **SimplicialCubature** (Nolan et al., 2016). It computes multivariate integrals over simplices. Hence, to compute an integral of a function over a convex polytope P in R, one should compute the Delaunay triangulation with package **geometry** and then use the package **SimplicialCubature** to sum the values of all the integrals over the simplices computed by the triangulation. The pattern is similar to volume computation. For $d = 5, 10$ the exact computation is faster than the approximate. For $d = 15$, **volesti** is 13 times faster. For $d = 20$, the exact approach halts, while **volesti** returns an estimation in less than a minute.

```
R> num_of_points = 5000
R> f = function(x) { sum(x^2) + (2 * x[1]^2 + x[2] + x[3]) }
R> for (d in seq(from = 5, to = 20, by = 5)) {
     P = gen_rand_vpoly(d, 2 * d, generator = list("seed" = 127))

     points = sample_points(P, random_walk = list("walk" = "BiW",
                            "walk_length" = 1, "seed" = 5), n = num_of_points)
     sum_f = 0
     for (i in seq_len(num_of_points)){
       sum_f = sum_f + f(points[, i])
     }
     V = volume(P, settings = list("error" = 0.05, "seed" = 5))
     I2 = (sum_f * V) / num_of_points
   }
```

Combinatorics and artificial intelligence

We focus now on a different problem, namely, counting the linear extensions of a given partially ordered set (poset), which arises in various applications in artificial intelligence and machine learning, such as partial order plans (Muise et al., 2016) and learning graphical models (Niinimäki et al., 2016).

Let $G = (V, E)$ be an acyclic digraph with $V = [n] := \{1, 2, \dots, n\}$. One might want to consider G as a representation of the poset $V : i > j$ if and only if there is a directed path from node i to node j. A permutation π of $[n]$ is called a linear extension of G (or the associated poset V) if $\pi^{-1}(i) > \pi^{-1}(j)$ for every edge $i \to j \in E$.

Let $P_{LE}(G)$ be the polytope in $\mathbb{R}^n$ defined by

$$P_{LE}(G) = \{x \in \mathbb{R}^n \mid 1 \geq x_i \geq 0 \text{ for all } i = 1, 2, \dots, n\},$$

and $x_i \geq x_j$ for all directed edges $i \to j \in E$. It is well known (Stanley, 1986) that the number of linear

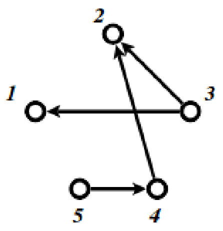


Figure 8: An acyclic directed graph with 5 nodes, 4 edges, and 9 linear extensions.

extensions of G equals the normalized volume of $P_{LE}(G)$, i.e.,

$$\#_{LE}G = \mathrm{vol}(P_{LE}(G))\, n!$$

It is also well known that counting linear extensions is #P-complete (Brightwell and Winkler, 1991). Thus, as the number of graph nodes (i.e., the dimension of $P_{LE}(G)$) grows, the problem becomes intractable for exact methods. Interestingly, **volesti** provides an efficient approximation method that could be added to the ones surveyed by Talvitie et al. (2018).

As a simple example, consider the graph in Figure 8 that has 9 linear extensions[1]. This number can be estimated in milliseconds using **volesti** as in the following script, where the estimated number of linear extensions is 9.014706.

```
R> A = matrix(c(
              -1,0,1,0,0,0,
              -1,1,0,0,0,-1,
              0,1,0,0,0,0,-1,
              1,1,0,0,0,0,0,
              1,0,0,0,0,0,1,
              0,0,0,0,0,1,0,
              0,0,0,0,1,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1,
              0,0,0,0,0,-1),
              ncol = 5, nrow = 14, byrow = TRUE)
R> b = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 , 1, 1, 1)
R> P_LE = Hpolytope(A = A, b = b)
R> time = system.time({ LE = volume(P_LE, settings = list("error" = 0.01,
                                    "seed" = 1927)) * factorial(5) })
```

Concluding remarks and future work

volesti is an R package that provides MCMC sampling routines for multivariate distributions restricted to convex polytopes and volume estimation. It supports three different polytope representations, and thus, it is useful for several applications. We illustrate the usage of **volesti** with simple, reproducible examples and show how **volesti** can be used to address challenging problems in modern applications.

Regarding future work, the expansion of **volesti** to support general log-concave sampling methods would be of special interest for several applications. Efficient log-concave sampling could also lead to additional sophisticated methods to estimate a multivariate integral over a convex polytope (Lovasz and Vempala, 2006).

Computational details

The results in this paper were obtained using R 3.4.4, R 3.6.3, and **volesti** 1.1.2-2. The versions of the imported by **volesti** packages are **stats** 3.4.4 (R Core Team, 2020b) and **methods** 3.4.4 (R Core Team, 2020a); of the linked by **volesti** packages, **Rcpp** 1.0.3, **BH** 1.69.0.1 (Eddelbuettel et al., 2020a), **RcppEigen** 0.3.3.7.0 (Bates and Eddelbuettel, 2013). The suggested package is **testthat** 2.0.1 (Wickham, 2011). For comparison to **volesti** and for plots, this paper uses **geometry** 0.4.5, **hitandrun** 0.5.5,

[1]Example taken from `https://people.inf.ethz.ch/fukudak/lect/pclect/notes2016/expoly_order.pdf`

SimplicialCubature 1.2, **Rfast** 2.0.3 (Papadakis et al., 2021), **ggplot2** 3.1.0 (Wickham, 2016), **plotly** 4.8.0 (Sievert, 2020), **rgl** 0.100.50 (Adler et al., 2021), **coda** 0.19.4. All packages used are available from CRAN.

All computations were performed on a PC with `Intel® Pentium(R) CPU G4400 @ 3.30GHz` × `2` `CPU` and `16GB RAM`.

Acknowledgments

The main part of the work has been done, while A.C. was supported by Google Summer of Code 2018 and 2019 grants, and V.F. was his mentor. The authors acknowledge fruitful discussions with Ioannis Emiris, Ludovic Calès, Elias Tsigaridas, the R-project for statistical computing, and the R community.

Bibliography

D. Adler, D. Murdoch, et al. *rgl: 3D Visualization Using OpenGL*, 2021. URL `https://CRAN.R-project.org/package=rgl`. R package version 0.105.13. [p656]

J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of The American Statistical Association - J AMER STATIST ASSN*, 88:669–679, 06 1993. URL `https://doi.org/10.1080/01621459.1993.10476321`. [p643]

M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, Aug 2014. ISSN 1552-3098. URL `https://doi.org/10.1109/TRO.2014.2312453`. [p653]

Y. Altmann, S. McLaughlin, and N. Dobigeon. Sampling from a multivariate gaussian distribution truncated on a simplex: A review. In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pages 113–116, 2014. URL `https://doi.org/10.1109/SSP.2014.6884588`. [p649]

J. M. Azaïs, S. Bercu, J. C. Fort, A. Lagnoux, and P. Lé. Simultaneous confidence bands in curve prediction applied to load curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(5):889–904, 2010. URL `https://doi.org/10.1111/j.1467-9876.2010.00727.x`. [p643]

A. Banerjee and C.-H. Hung. Informed momentum trading versus uninformed "naive" investors strategies. *Journal of Banking & Finance*, 35(11):3077–3089, 2011. ISSN 0378-4266. URL `https://doi.org/10.1016/j.jbankfin.2011.04.005`. [p652]

C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, Dec. 1996. ISSN 0098-3500. URL `https://doi.org/10.1145/235815.235821`. [p643]

D. Bates and D. Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. URL `http://www.jstatsoft.org/v52/i05/`. [p655]

M. Berkelaar, K. Eikland, and P. Notebaert. **lp_solve** 5.5, open source (mixed-integer) linear programming system. Software, May 1 2004. URL `http://lpsolve.sourceforge.net/5.5/`. [p646]

D. Bolin and F. Lindgren. Excursion and contour uncertainty regions for latent gaussian models. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 77:85–106, 01 2015. URL `https://doi.org/10.1111/rssb.12055`. [p643]

G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8:225–242, 1991. URL `https://doi.org/10.1007/BF00383444`. [p655]

L. Calès, A. Chalkis, I. Emiris, and V. Fisikopoulos. Practical Volume Computation of Structured Convex Bodies, and an Application to Modeling Portfolio Dependencies and Financial Crises. In B. Speckmann and C. D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 19:1–19:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL `https://doi.org/10.4230/LIPIcs.SoCG.2018.19`. [p642, 643, 651, 652]

A. Chalkis and V. Fisikopoulos. Blog post on sampling, integration and volume computations in R, 2021. URL `https://geomscale.github.io/Sampling-integration-volume-in-R`. [p642, 647, 649, 650]

A. Chalkis, I. Z. Emiris, and V. Fisikopoulos. Practical volume estimation by a new annealing schedule for cooling convex bodies, 2019. URL http://arxiv.org/abs/1905.05494. [p642, 646]

Y. Chen, R. Dwivedi, M. Wainwright, and B. Yu. Fast MCMC sampling algorithms on polytopes. *Journal of Machine Learning Research*, 19(55):1–86, 2018. URL http://jmlr.org/papers/v19/18-158.html. [p642]

C. Chivers. **MHadaptive**: *General Markov Chain Monte Carlo for Bayesian Inference using adaptive Metropolis-Hastings sampling*, 2012. R package version 1.1-8. [p643]

B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8 (2), Jun 2016. URL https://doi.org/10.1007/s12532-015-0097-z. [p642]

K. V. den Meersche, K. Soetaert, and D. V. Oevelen. xsample(): An r function for sampling linear inverse problems. *Journal of Statistical Software, Code Snippets*, 30(1):1–15, 2009. ISSN 1548-7660. doi: 10.18637/jss.v030.c01. URL https://www.jstatsoft.org/v030/c01. [p643]

V. den Meersche K., S. K., and van Oevelen D. **limSolve**: *Solving Linear Inverse Models*, 2009. URL https://CRAN.R-project.org/package=limSolve. R package version 1.5.1. [p643]

M. Duca, A. Koban, M. Basten, E. Bengtsson, B. Klaus, P. Kusmierczyk, J. Lang, C. Detken, and T. Peltonen. A new Database for Financial Crises in European Countries. Occasional Paper Series 194, European Central Bank, July 2017. URL https://ideas.repec.org/p/ecb/ecbops/2017194.html. [p652]

M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988. URL https://doi.org/10.1137/0217060. [p642]

D. Eddelbuettel, J. W. Emerson, and M. J. Kane. *BH: Boost C++ Header Files*, 2020a. URL https://CRAN.R-project.org/package=BH. R package version 1.72.0-3. [p655]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. **Rcpp**: *Seamless R and C++ Integration*, 2020b. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.5. [p646]

I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Transactions of Mathematical Software, 2018*, 44(4):38:1–38:21, 2014. ISSN 0098-3500. URL https://doi.org/10.1145/3194656. [p642, 646]

V. Fisikopoulos, A. Chalkis, and contributors in file inst/AUTHORS. *volesti: Volume Approximation and Sampling of Convex Polytopes*, 2020. R package version 1.1.2. [p642]

A. Genz and F. Bretz. *Computation of Multivariate Normal and t Probabilities*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 364201688X, 9783642016882. [p642]

C. Geyer. Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022:45, 2011. [p647, 651]

C. J. Geyer and L. T. Johnson. **mcmc**: *Markov Chain Monte Carlo*, 2020. R package version 0.9-7. [p643]

B. Grün and K. Hornik. Modelling human immunodeficiency virus ribonucleic acid levels with finite mixtures for censored longitudinal data. *Journal of the Royal Statistical Society. Series C, Applied statistics*, 61:201–218, 03 2012. URL https://doi.org/10.1111/j.1467-9876.2011.01007.x. [p643]

D. Guegan, L. Calès, and M. Billio. A Cross-Sectional Score for the Relative Performance of an Allocation. Université Paris 1 Panthéon-Sorbonne (Post-Print and Working Papers) halshs-00646070, HAL, 2011. URL https://ideas.repec.org/p/hal/cesptp/halshs-00646070.html. [p652]

G. Guennebaud, B. Jacob, et al. *Eigen v3*, 2010. URL http://eigen.tuxfamily.org. [p646]

W. Hallerbach, C. Hundack, I. Pouchkarev, and J. Spronk. A broadband vision of the development of the dax over time. Technical Report ERS-2002-87-F&A, Erasmus University Rotterdam, 2002. [p643]

H. Haraldsdòttir, B. Cousins, I. Thiele, R. Fleming, and S. Vempala. CHRR: Coordinate Hit-and-Run with Rounding for Uniform Sampling of Constraint-Based Models. *Bioinformatics*, 33(11):1741–1743, 01 2017. URL https://doi.org/10.1093/bioinformatics/btx052. [p643, 646]

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL http://www.jstor.org/stable/2334940. [p642, 645]

D. W. Heck. **multinomineq**: *Bayesian Inference for Multinomial Models with Inequality Constraints*, 2019. R package version 0.2.1. [p643]

R. Huser and A. Davison. Composite likelihood estimation for the brown-resnick process. *Biometrika*, 2, 06 2013. URL https://doi.org/10.1093/biomet/ass089. [p643]

S. Iyengar. Evaluation of normal probabilities of symmetric regions. *SIAM Journal on Scientific and Statistical Computing*, 9(3):418–423, 1988. URL https://doi.org/10.1137/0909028. [p642]

M. C. Jensen. The Performance of Mutual Funds in the Period 1945-1964. SSRN Scholarly Paper ID 244153, Social Science Research Network, Rochester, NY, May 1967. URL https://papers.ssrn.com/abstract=244153. [p652]

A. Kopetzki, B. Schürmann, and M. Althoff. Methods for order reduction of zonotopes. In *IEEE Conference on Decision and Control*, pages 5626–5633, 2017. URL https://doi.org/10.1109/CDC.2017.8264508. [p642, 653]

A. Laddha and S. Vempala. Convergence of Gibbs Sampling: Coordinate Hit-and-Run Mixes Fast, 2020. [p645]

Y. Lee and S. Vempala. Eldan's stochastic localization and the KLS hyperplane conjecture: An improved lower bound for expansion. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 998–1007, 2017. URL https://doi.org/10.1109/FOCS.2017.96. [p645, 646]

Y. Lee and S. Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1115–1121, 2018. ISBN 978-1-4503-5559-9. URL https://doi.org/10.1145/3188745.3188774. [p642]

A. F. Lopez. **lineqGPR**: *Gaussian Process Regression Models with Linear Inequality Constraints*, 2019. R package version 0.1.1. [p643]

L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences*, 72:392–417, 2006. URL https://doi.org/10.1016/j.jcss.2005.08.004. [p646]

L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006. ISSN 0097-5397. URL https://doi.org/10.1137/S009753970544727X. [p642, 645]

L. Lovasz and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 57–68, 2006. doi: 10.1109/FOCS.2006.28. [p655]

L. Lovász, R. Kannan, and M. Simonovits. Random walks and an $o^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997. URL https://doi.org/10.1002/(SICI)1098-2418(199708)11:1<1::AID-RSA1>3.0.CO;2-X. [p646]

T. F. R. Ma, S. K. Ghosh, and Y. Li. **tmvmixnorm**: *Sampling from Truncated Multivariate Normal and t Distributions*, 2020. R package version 1.1.1. [p643]

O. Mangoubi and N. K. Vishnoi. Faster polytope rounding, sampling, and volume computation via a sub-linear ball walk. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1338–1357, 2019. URL https://doi.org/10.1109/FOCS.2019.00082. [p642]

J. Maurer and S. Watanabe. Boost random number library. Software, 2017. URL https://www.boost.org/doc/libs/1_73_0/doc/html/boost_random.html. [p646]

R. D. Morey. **HybridMC**: *Implementation of the Hybrid Monte Carlo and Multipoint Hybrid Monte Carlo sampling techniques*, 2009. R package version 0.2. [p643]

C. Muise, J. Beck, and S. McIlraith. Optimal partial-order plan relaxation via maxsat. *Journal of Artificial Intelligence Research*, 57:113–149, 09 2016. URL https://doi.org/10.1613/jair.5128. [p654]

T. Niinimäki, P. Parviainen, and M. Koivisto. Structure discovery in bayesian networks by sampling partial orders. *Journal of Machine Learning Research*, 17(57):1–47, 2016. URL http://jmlr.org/papers/v17/15-140.html. [p654]

J. P. Nolan, with parts adapted from Fortran, and matlab code by Alan Genz. **SimplicialCubature**: *Integration of Functions Over Simplices*, 2016. URL https://CRAN.R-project.org/package=SimplicialCubature. R package version 1.2. [p654]

A. Pakman. **tmg**: *Truncated Multivariate Gaussian Sampling*, 2015. R package version 0.3. [p643]

M. Papadakis, M. Tsagris, M. Dimitriadis, S. Fafalios, I. Tsamardinos, M. Fasiolo, G. Borboudakis, J. Burkardt, C. Zou, K. Lakiotaki, and C. Chatzipantsiou. *Rfast: A Collection of Efficient and Extremely Fast R Functions*, 2021. URL https://CRAN.R-project.org/package=Rfast. R package version 2.0.3. [p656]

A. Pereira and M. Althoff. Safety control of robots under computed torque control using reachable sets. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 331–338, May 2015. URL https://doi.org/10.1109/ICRA.2015.7139020. [p653]

M. Plummer, N. Best, K. Cowles, K. Vines, D. Sarkar, D. Bates, R. Almond, and A. Magnusson. **coda**: *Output Analysis and Diagnostics for MCMC*, 2020. URL https://CRAN.R-project.org/package=coda. R package version 0.19-4. [p647]

B. Polyak and E. Gryazina. Billiard walk - a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes*, 47(3):6123 – 6128, 2014. ISSN 1474-6670. URL https://doi.org/10.3182/20140824-6-ZA-1003.02312. 19th IFAC World Congress. [p642, 645, 646]

I. Pouchkarev. *Performance Evaluation of Constrained Portfolios*. PhD thesis, Erasmus Research Institute of Management, The Netherlands, 2005. [p652]

I. Pouchkarev, J. Spronk, and J. Trinidad. Dynamics of the spanish stock market through a broadband view of the IBEX 35 index. *Estudios Econom. Aplicada*, 22(1):7–21, 2004. [p643]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020a. URL https://www.R-project.org/. [p655]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020b. URL https://www.R-project.org/. [p655]

J.-R. Roussel, C. B. Barber, K. Habel, R. Grasman, R. B. Gramacy, P. Mozharovskyi, and D. C. Sterratt. **geometry**: *Mesh Generation and Surface Tessellation*, 2019. R package version 0.4.5. [p643]

V. Sartório. **rhmc**: *Hamiltonian Monte Carlo*, 2018. R package version 1.0.0. [p643]

J. Schellenberger and B. Palsson. Use of randomized sampling for analysis of metabolic networks. *The Journal of biological Chemistry*, 284 9:5457–61, 2009. URL https://doi.org/10.1074/jbc.R800048200. [p642]

W. F. Sharpe. Mutual Fund Performance. *The Journal of Business*, 39(1):119–138, 1966. ISSN 0021-9398. URL https://www.jstor.org/stable/2351741. [p652]

C. Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN 9781138331457. URL https://plotly-r.com. [p656]

R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984. ISSN 0030364X, 15265463. URL http://www.jstor.org/stable/170949. [p642, 644]

P. Somerville. Numerical computation of multivariate normal and multivariate-t probabilities over convex regions. *Journal of Computational and Graphical Statistics*, 7(4):529–544, 1998. URL https://doi.org/10.1080/10618600.1998.10474793. [p642]

R. P. Stanley. Two poset polytopes. *Discrete & Computational Geometry*, 1(1):9–23, Mar. 1986. ISSN 1432-0444. URL https://doi.org/10.1007/BF02187680. [p654]

T. Talvitie, K. Kangas, T. Niinimäki, and M. Koivisto. Counting linear extensions in practice: Mcmc versus exponential monte carlo. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16957. [p655]

J. Taylor and Y. Benjamini. **restrictedMVN**: *Multivariate Normal Restricted by Affine Constraints*, 2016. R package version 1.0. [p643]

J. L. Treynor. How to Rate Management of Investment Funds. In *Treynor on Institutional Investing*, pages 69–87. John Wiley & Sons, Ltd, 2015. ISBN 978-1-119-19667-9. URL https://doi.org/10.1002/9781119196679.ch10. [p652]

G. van Valkenhoef and T. Tervonen. **hitandrun**: *"Hit and Run" and "Shake and Bake" for Sampling Uniformly from Convex Shapes*, 2019. URL https://CRAN.R-project.org/package=hitandrun. R package version 0.5-5. [p643]

G. Varsi. The multidimensional content of the frustum of the simplex. *Pacific Journal of Mathematics*, 46 (1):303–314, 1973. URL https://projecteuclid.org:443/euclid.pjm/1102946623. [p652]

A. Venzke, D. Molzahn, and S. Chatzivasileiadis. Efficient creation of datasets for data-driven power system applications. *Electric Power Systems Research*, 190:106614, 2021. ISSN 0378-7796. URL https://doi.org/10.1016/j.epsr.2020.106614. [p642]

J. Wadsworth and J. Tawn. Efficient inference for spatial extreme value processes associated to log-gaussian random functions. *Biometrika*, 1, 03 2014. URL https://doi.org/10.1093/biomet/ast042. [p643]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p655]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p656]

G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, New York, 1995. URL https://doi.org/10.1007/978-1-4613-8431-1. [p644]

Apostolos Chalkis
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens
Greece
GeomScale org.
ORCiD: 0000-0002-4628-1907
achalkis@di.uoa.gr

Vissarion Fisikopoulos
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens
Greece
GeomScale org.
ORCiD: 0000-0002-0780-666X
vfisikop@di.uoa.gr

Elliptical Symmetry Tests in R

by Slađana Babić, Christophe Ley and Marko Palangetić

Abstract The assumption of elliptical symmetry has an important role in many theoretical developments and applications. Hence, it is of primary importance to be able to test whether that assumption actually holds true or not. Various tests have been proposed in the literature for this problem. To the best of our knowledge, none of them has been implemented in R. This article describes the R package **ellipticalsymmetry** which implements several well-known tests for elliptical symmetry together with some recent tests. We demonstrate the testing procedures with a real data example.

Introduction

Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ denote a sample of n i.i.d. d-dimensional observations. A d-dimensional random vector $\mathbf{X}$ is said to be elliptically symmetric about some location parameter $\boldsymbol{\theta} \in R^d$ if its density $\underline{f}$ is of the form

$$\mathbf{x} \mapsto \underline{f}(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\Sigma},f) = c_{d,f}|\boldsymbol{\Sigma}|^{-1/2} f\left(\|\boldsymbol{\Sigma}^{-1/2}(\mathbf{x}-\boldsymbol{\theta})\|\right), \qquad \mathbf{x} \in \mathbb{R}^d, \tag{1}$$

where $\boldsymbol{\Sigma} \in \mathcal{S}_d$ (the class of symmetric positive definite real $d \times d$ matrices) is a *scatter* parameter, $f : \mathbb{R}_0^+ \to \mathbb{R}^+$ is an, a.e., strictly positive function called *radial density*, and $c_{d,f}$ is a normalizing constant depending on f and the dimension d. Many well-known and widely used multivariate distributions are elliptical. The multivariate normal, multivariate Student t, multivariate power-exponential, symmetric multivariate stable, symmetric multivariate Laplace, multivariate logistic, multivariate Cauchy, and multivariate symmetric general hyperbolic distribution are all examples of elliptical distributions. The family of elliptical distributions has several appealing properties. For instance, it has a simple stochastic representation, clear parameter interpretation, it is closed under affine transformations, and its marginal and conditional distributions are also elliptically symmetric; see Paindaveine (2014) for details. Thanks to its mathematical tractability and nice properties, it became a fundamental assumption in multivariate analysis and many applications. Numerous statistical procedures, therefore, rest on the assumption of elliptical symmetry: one- and K-sample location and shape problems (Um and Randles, 1998; Hallin and Paindaveine, 2002, 2006; Hallin et al., 2006), serial dependence and time series (Hallin and Paindaveine, 2004), one- and K-sample principal component problems (Hallin et al., 2010, 2014), multivariate tail estimation (Dominicy et al., 2017), to cite but a few. Elliptical densities are also considered in portfolio theory (Owen and Rabinovitch, 1983), capital asset pricing models (Hodgson et al., 2002), semiparametric density estimation (Liebscher, 2005), graphical models (Vogel and Fried, 2011), and many other areas.

Given the omnipresence of the assumption of elliptical symmetry, it is essential to be able to test whether that assumption actually holds true or not for the data at hand. Numerous tests have been proposed in the literature, including Beran (1979), Baringhaus (1991), Koltchinskii and Sakhanenko (2000), Manzotti et al. (2002), Schott (2002), Huffer and Park (2007), Cassart (2007), and Babić et al. (2021). Tests for elliptical symmetry based on Monte Carlo simulations can be found in Diks and Tong (1999) and Zhu and Neuhaus (2000); Li et al. (1997) recur to graphical methods, and Zhu and Neuhaus (2004) build conditional tests. We refer the reader to Serfling (2006) and Sakhanenko (2008) for extensive reviews and performance comparisons. To the best of our knowledge, none of these tests is available in the open software R. The focus of this paper is to close this gap by implementing several well-known tests for elliptical symmetry together with some recent tests. The test of Beran (1979) is neither distribution-free nor affine-invariant. Moreover, there are no practical guidelines to the choice of the basis functions involved in the test statistic. Therefore, we opt not to include it in the package. Baringhaus (1991) proposes a Cramér-von Mises type test for spherical symmetry based on the independence between norm and direction. Dyckerhoff et al. (2015) have shown by simulations that this test can be used as a test for elliptical symmetry in dimension 2. This test assumes the location parameter to be known and its asymptotic distribution is not simple to use (plus no proven validity in dimensions higher than 2). Hence, we decided not to include it in the package. Thus, the tests suggested by Koltchinskii and Sakhanenko (2000), Manzotti et al. (2002), Schott (2002), Huffer and Park (2007), Cassart (2007), and Babić et al. (2021) are implemented in the package **ellipticalsymmetry**.

This paper describes the tests for elliptical symmetry that have been implemented in the **ellipticalsymmetry** package, together with a detailed description of the functions that are available in the package. The use of the implemented functions is illustrated using financial data.

Testing for elliptical symmetry

In this section, we focus on the tests for elliptical symmetry that have been implemented in our new **ellipticalsymmetry** package. Besides formal definitions of test statistics and limiting distributions, we also explain the details of computation.

Test by Koltchinskii and Sakhanenko

Koltchinskii and Sakhanenko (2000) develop a class of omnibus bootstrap tests for unspecified location that are affine invariant and consistent against any fixed alternative. The estimators of the unknown parameters are as follows: $\hat{\boldsymbol{\theta}} = n^{-1}\sum_{i=1}^{n}\mathbf{X}_i$ and $\hat{\boldsymbol{\Sigma}} = n^{-1}\sum_{i=1}^{n}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})(\mathbf{X}_i - \hat{\boldsymbol{\theta}})'$. Define $\mathbf{Y}_i = \hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})$ and let $\mathcal{F}_B$ be a class of Borel functions from $\mathbb{R}^d$ to R. Their test statistics are functionals (for example, sup-norms) of the stochastic process

$$n^{-1/2}\sum_{i=1}^{n}\left(f(\mathbf{Y}_i) - m_f(||\mathbf{Y}_i||)\right),$$

where $f \in \mathcal{F}_B$ and $m_f(\rho)$ is the average value of f on the sphere with radius $\rho > 0$. Several examples of classes $\mathcal{F}_B$ and test statistics based on the sup-norm of the above process are considered in Koltchinskii and Sakhanenko (2000). Here, we restrict our attention to $\mathcal{F}_B := \left\{ I_{0<||\mathbf{x}||\leq t}\psi\left(\frac{\mathbf{x}}{||\mathbf{x}||}\right) : \psi \in G_l, ||\psi||_2 \leq 1, t > 0\right\}$, where I_A stands for the indicator function of A, G_l for the linear space of spherical harmonics of degree less than or equal to l in R^d, and $||\cdot||_2$ is the L^2-norm on the unit sphere $\mathcal{S}^{d-1}$ in R^d. With these quantities in hand, the test statistic becomes

$$Q_{KS}^{(n)} := n^{-1/2}\max_{1\leq j\leq n}\left(\sum_{s=1}^{\dim(G_l)}\left(\sum_{k=1}^{j}\psi_s\left(\frac{\mathbf{Y}_{[k]}}{||\mathbf{Y}_{[k]}||}\right) - \delta_{s1}\right)^2\right)^{1/2},$$

where $\mathbf{Y}_{[i]}$ denotes the ith order statistic from the sample $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$, ordered according to their L^2-norm, $\{\psi_s,\ s = 1, \ldots, \dim(G_l)\}$ denotes an orthonormal basis of G_l, $\psi_1 = 1$, and $\delta_{ij} = 1$ for $i = j$ and 0 otherwise. The test statistic is relatively simple to construct if we have formulas for spherical harmonics. In dimension 2, spherical harmonics coincide with sines and cosines on the unit circle. The detailed construction of the test statistic $Q_{KS}^{(n)}$ for dimensions 2 and 3 can be found in Sakhanenko (2008). In order to be able to use $Q_{KS}^{(n)}$ in higher dimensions, we need corresponding formulas for spherical harmonics. Using recursive formulas from Müller (1966) and equations given in Manzotti and Quiroz (2001), we obtained spherical harmonics of degree one to four in arbitrary dimension. The reader should bare in mind that the larger degree leads to the better power performance of this test. A drawback of this test is that it requires bootstrap procedures to obtain critical values.

In our R package, this test can be run using a function called `KoltchinskiiSakhanenko()`. The syntax for this function is very simple:

```
KoltchinskiiSakhanenko(X, R=1000, nJobs = -1),
```

where `X` is an input to this function consisting of a data set which must be a matrix, and `R` stands for the number of bootstrap replicates. The default number of replicates is set to 1000. The `nJobs` argument represents the number of CPU cores to use for the calculation. This is a purely technical option which is used to speed up the computation of bootstrap-based tests. The default value `-1` indicates that all cores except one are used.

The MPQ test

Manzotti et al. (2002) develop a test based on spherical harmonics. The estimators of the unknown parameters are the sample mean denoted as $\hat{\boldsymbol{\theta}}$, and the unbiased sample covariance matrix given by $\hat{\boldsymbol{\Sigma}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})(\mathbf{X}_i - \hat{\boldsymbol{\theta}})'$. Define, again, $\mathbf{Y}_i = \hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})$. When the $\mathbf{X}_i'$s are elliptically symmetric, then $\mathbf{Y}_i/||\mathbf{Y}_i||$ should be uniformly distributed on the unit sphere. Manzotti et al. (2002) chose this property as the basis of their test. The uniformity of the standardized vectors $\mathbf{Y}_i/||\mathbf{Y}_i||$ can be checked in different ways. Manzotti et al. (2002) opt to verify this uniformity using spherical harmonics. For a fixed $\varepsilon > 0$, let ρ_n be the ε sample quantile of $||\mathbf{Y}_1||, \ldots, ||\mathbf{Y}_n||$. Then, the test statistic is

$$Q_{MPQ}^{(n)} = n\sum_{h\in\mathcal{S}_{jl}}\left(\frac{1}{n}\sum_{i=1}^{n}h\left(\frac{\mathbf{Y}_i}{||\mathbf{Y}_i||}\right)I(||\mathbf{Y}_i|| > \rho_n)\right)^2$$

for $l \geq j \geq 3$, where $\mathcal{S}_{jl} = \bigcup_{j\leq i\leq l}\mathcal{H}_i$ and $\mathcal{H}_i$ is the set of spherical harmonics of degree i. In the implementation of this test we used spherical harmonics of degree 3 and 4. The asymptotic distribution of the test statistic $Q^{(n)}_{MPQ}$ is $(1-\varepsilon)\chi$, where χ is a variable with a chi-squared distribution with ν_{jl} degrees of freedom, where ν_{jl} denotes the total number of functions in $\mathcal{S}_{jl}$. Note that $Q^{(n)}_{MPQ}$ is only a necessary condition statistic for the null hypothesis of elliptical symmetry, and therefore, this test does not have asymptotic power against all alternatives. In the **ellipticalsymmetry** package, this test is implemented as the `MPQ()` function with the following syntax

```
MPQ(X, epsilon = 0.05).
```

As before, `X` is a numeric matrix that represents the data while `epsilon` is an option that allows the user to indicate the proportion of points $\mathbf{Y}_i$ close to the origin which will not be used in the calculation. By doing this, extra assumptions on the radial density in (1) are avoided. The default value of `epsilon` is set to 0.05.

Schott's test

Schott (2002) develops a Wald-type test for elliptical symmetry based on the analysis of covariance matrices. The test compares the sample fourth moments with the expected theoretical ones under ellipticity. Given that the test statistic involves consistent estimates of the covariance matrix of the sample fourth moments, the existence of eight-order moments is required. Furthermore, the test has a very low power against several alternatives. The final test statistic is of a simple form, even though it requires lengthy notations.

For an elliptical distribution with mean $\boldsymbol{\theta}$ and covariance matrix $\boldsymbol{\Sigma}$, the fourth moment defined as $\boldsymbol{M}_4 = E\{(\mathbf{X}-\boldsymbol{\theta})(\mathbf{X}-\boldsymbol{\theta})' \otimes (\mathbf{X}-\boldsymbol{\theta})(\mathbf{X}-\boldsymbol{\theta})'\}$, with $\otimes$ the Kronecker product, has the form

$$\boldsymbol{M}_4 = (1+\kappa)((\mathbf{I}_{d^2}+K_{dd})(\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma}) + vec(\boldsymbol{\Sigma})vec(\boldsymbol{\Sigma})'), \tag{2}$$

where K_{dd} is a commutation matrix (Magnus, 1988), $\mathbf{I}_d$ is the $d\times d$ identity matrix, and κ is a scalar, which can be expressed using the characteristic function of the elliptical distribution. Here, the *vec* operator stacks all components of a $d\times d$ matrix $\boldsymbol{M}$ on top of each other to yield the d^2 vector vec$(\boldsymbol{M})$. Let $\hat{\boldsymbol{\Sigma}}$ denotes the usual unbiased sample covariance matrix and $\hat{\boldsymbol{\theta}}$ the sample mean. A simple estimator of $\boldsymbol{M}_4$ is given by $\hat{\boldsymbol{M}}_4 = \frac{1}{n}\sum_{i=1}^n(\mathbf{X}_i-\hat{\boldsymbol{\theta}})(\mathbf{X}_i-\hat{\boldsymbol{\theta}})'\otimes(\mathbf{X}_i-\hat{\boldsymbol{\theta}})(\mathbf{X}_i-\hat{\boldsymbol{\theta}})'$, and its standardized version is given by

$$\hat{\boldsymbol{M}}_{4*} = (\hat{\boldsymbol{\Sigma}}^{-1/2'}\otimes\hat{\boldsymbol{\Sigma}}^{-1/2'})\hat{\boldsymbol{M}}_4(\hat{\boldsymbol{\Sigma}}^{-1/2}\otimes\hat{\boldsymbol{\Sigma}}^{-1/2}).$$

Then, an estimator of $vec(\boldsymbol{M}_4)$ is constructed as $\boldsymbol{G} = vec(\boldsymbol{N}_4)vec(\boldsymbol{N}_4)'vec(\hat{\boldsymbol{M}}_{4*})/(3d(d+2))$, and it is consistent if and only if $\boldsymbol{M}_4$ is of the form (2). Here, $\boldsymbol{N}_4$ represents the value of $\boldsymbol{M}_4$ under the multivariate standard normal distribution. Note that the asymptotic mean of $\boldsymbol{v} = n^{1/2}(vec(\hat{\boldsymbol{M}}_{4*})-\boldsymbol{G})$ is 0 if and only if (2) holds, and this expression is used to construct the test statistic. Denote the estimate of the asymptotic covariance matrix of $n^{1/2}\boldsymbol{v}$ as $\hat{\boldsymbol{\Phi}}$. The Wald test statistic is then formalized as $T = \boldsymbol{v}'\hat{\boldsymbol{\Phi}}^{-}\boldsymbol{v}$, where $\hat{\boldsymbol{\Phi}}^{-}$ is a generalized inverse of $\hat{\boldsymbol{\Phi}}$. For more technical details, we refer the reader to Section 2 in Schott (2002). In order to define Schott's test statistic, we further have to define the following quantities:

$$(1+\hat{\kappa}) = \frac{1}{nd(d+2)}\sum_{i=1}^{n}\{(\mathbf{X}_i-\hat{\boldsymbol{\theta}})'\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{X}_i-\hat{\boldsymbol{\theta}})\}^2$$

$$(1+\hat{\eta}) = \frac{1}{nd(d+2)(d+4)}\sum_{i=1}^{n}\{(\mathbf{X}_i-\hat{\boldsymbol{\theta}})'\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{X}_i-\hat{\boldsymbol{\theta}})\}^3$$

$$(1+\hat{\omega}) = \frac{1}{nd(d+2)(d+4)(d+6)}\sum_{i=1}^{n}\{(\mathbf{X}_i-\hat{\boldsymbol{\theta}})'\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{X}_i-\hat{\boldsymbol{\theta}})\}^4.$$

Moreover, let $\hat{\beta}_1 = (1+\hat{\omega})^{-1}/24$, $\hat{\beta}_2 = -3a\{24(1+\hat{\omega})^2+12(d+4)a(1+\hat{\omega})\}^{-1}$, $a = (1+\hat{\omega})+(1+\hat{\kappa})^3-2(1+\hat{\kappa})(1+\hat{\eta})$. Finally, the test statistic becomes

$$T = n\left[\hat{\beta}_1\mathrm{tr}(\hat{\boldsymbol{M}}_{4*}^2)+\hat{\beta}_2\mathrm{vec}(\mathbf{I}_d)'\hat{\boldsymbol{M}}_{4*}^2\mathrm{vec}(\mathbf{I}_d)-\{3\hat{\beta}_1+(d+2)\hat{\beta}_2\}d(d+2)(1+\hat{\kappa})^2\right].$$

It has an asymptotic chi-squared distribution with degrees of freedom $\nu_d = d^2+\dfrac{d(d-1)(d^2+7d-6)}{24}-1$.

The Schott test can be performed in our package by using the function `Schott()` with the very simple syntax `Schott(X)`, where `X` is a numeric matrix of data values.

Test by Huffer and Park

Huffer and Park (2007) propose a Pearson chi-square type test with multi-dimensional cells. Under the null hypothesis of ellipticity, the cells have asymptotically equal expected cell countsm, and after determining the observed cell counts, the test statistic is easily computed. Let $\hat{\boldsymbol{\theta}}$ be the sample mean and $\hat{\boldsymbol{\Sigma}} = n^{-1}\sum_{i=1}^{n}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})(\mathbf{X}_i - \hat{\boldsymbol{\theta}})'$ the sample covariance matrix. Define $\mathbf{Y}_i = \mathbf{R}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})$, where the matrix $\mathbf{R} = \mathbf{R}(\hat{\boldsymbol{\Sigma}})$ is a function of $\hat{\boldsymbol{\Sigma}}$ such that $\mathbf{R}\hat{\boldsymbol{\Sigma}}\mathbf{R} = \mathbf{I}_d$. Typically $\mathbf{R} = \hat{\boldsymbol{\Sigma}}^{-1/2}$ as for the previous tests. However, Huffer and Park suggest to use the Gram-Schmidt transformation because that will lead to standardized data whose joint distribution does not depend on $\boldsymbol{\theta}$ or $\boldsymbol{\Sigma}$. In order to compute the test statistic, the space $\mathbb{R}^d$ should be divided into c spherical shells centered at the origin such that each shell contains an equal number of the scaled residuals $\mathbf{Y}_i$. The next step is to divide $\mathbb{R}^d$ into g sectors such that for any pair of sectors, there is an orthogonal transformation mapping one onto the other. Therefore, the c shells and g sectors divide $\mathbb{R}^d$ into gc cells, which, under elliptical symmetry, should contain $n/(gc)$ of the vectors $\mathbf{Y}_i$. The test statistic then has the simple form

$$HP_n = \sum_{\pi}(U_\pi - np)^2/(np),$$

where U_π are cell counts for $\pi = (i,j)$ with $1 \le i \le g$ and $1 \le j \le c$ and $p = 1/(gc)$.

In the R package, we are considering three particular ways to partition the space: using (i) the 2^d orthants, (ii) permutation matrices, and (iii) a cyclic group consisting of rotations by angles which are multiples of $2\pi/g$. The first two options can be used for any dimension, while the angles are supported only for dimension 2. Huffer and Park's test can be run using a function called `HufferPark()`. The syntax, including all options, for the function `HufferPark()` is, for instance,

```
HufferPark(X, c, R = NA, sector = "orthants", g = NA, nJobs = -1).
```

We will now provide a detailed description of its arguments. `X` is an input to this function consisting of a data set. `sector` is an option that allows the user to specify the type of sectors used to divide the space. Currently supported options are `"orthants"`, `"permutations"`, and `"bivariateangles"`. The last one being available only in dimension 2. The `g` argument indicates the number of sectors. The user has to choose `g` only if `sector = "bivariateangles"` and it denotes the number of regions used to divide the plane. In this case, regions consist of points whose angle in polar coordinates is between $2(m-1)\pi/g$ and $2m\pi/g$ for $m \in \{1 \ldots g\}$. If `sector` is set to `"orthants"`, then `g` is fixed and equal to 2^d, while for `sector = "permutations"`, `g` is $d!$. No matter what type of sectors is chosen, the user has to specify the number of spherical shells that are used to divide the space, which is `c`. The value of `c` should be such that the average cell counts $n/(gc)$ are not too small. Several settings with different sample size and different values of g and c can be found in the simulation studies presented in Sections 4 and 5 of Huffer and Park (2007). As before, `nJobs` represents the number of CPU cores to use for the calculation. The default value `-1` indicates that all cores except one are used.

The asymptotic distribution is available only under `sector = "orthants"` when the underlying distribution is close to normal. It is a linear combination of chi-squared random variables, and it depends on eigenvalues of congruent sectors used to divide the space $\mathbb{R}^d$. Otherwise, bootstrap procedures are required, and the user can freely choose the number of bootstrap replicates, denoted as `R`. Note that by default, `sector` is set to `"orthants"` and `R = NA`.

Pseudo-Gaussian test

Cassart (2007) and Cassart et al. (2008) construct Pseudo-Gaussian tests for specified and unspecified location that are most efficient against a multivariate form of Fechner-type asymmetry (defined in Cassart (2007), Chapter 3). These tests are based on Le Cam's asymptotic theory of statistical experiments. We start by describing the specified-location Pseudo-Gaussian test. The unknown parameter $\boldsymbol{\Sigma}$ is estimated by using Tyler (1987)'s estimator of scatter which we simply denote by $\hat{\boldsymbol{\Sigma}}$. Let $m_k(\boldsymbol{\theta},\boldsymbol{\Sigma}) := n^{-1}\sum_{i=1}^{n}(\|\boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_i - \boldsymbol{\theta})\|)^k$, $U_i(\boldsymbol{\theta},\boldsymbol{\Sigma}) := \frac{\boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_i-\boldsymbol{\theta})}{\|\boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_i-\boldsymbol{\theta})\|}$ and

$$\mathbf{S}_i^{\mathbf{U}}(\boldsymbol{\theta},\boldsymbol{\Sigma}) := ((\mathbf{U}_{i1}(\boldsymbol{\theta},\boldsymbol{\Sigma}))^2\mathrm{sign}(\mathbf{U}_{i1}(\boldsymbol{\theta},\boldsymbol{\Sigma})),\ldots,(\mathbf{U}_{id}(\boldsymbol{\theta},\boldsymbol{\Sigma}))^2\mathrm{sign}(\mathbf{U}_{id}(\boldsymbol{\theta},\boldsymbol{\Sigma})))'.$$

The test statistic then has the simple form

$$Q_{p\mathcal{G},\boldsymbol{\theta}}^{(n)} = \frac{d(d+2)}{3nm_4(\boldsymbol{\theta},\hat{\boldsymbol{\Sigma}})}\sum_{i,j=1}^{n}(\|\hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_i-\boldsymbol{\theta})\|)^2(\|\hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_j-\boldsymbol{\theta})\|)^2\mathbf{S}_i'^{\mathbf{U}}(\boldsymbol{\theta},\hat{\boldsymbol{\Sigma}})\mathbf{S}_j^{\mathbf{U}}(\boldsymbol{\theta},\hat{\boldsymbol{\Sigma}}),$$

and follows asymptotically a chi-squared distribution χ_d^2 with d degrees of freedom. Finite moments of order four are required.

In most cases, the assumption of a specified center is, however, unrealistic. Cassart (2007), therefore, proposes also a test for the scenario when the location is not specified. The estimator of the unknown $\boldsymbol{\theta}$ is the sample mean denoted by $\hat{\boldsymbol{\theta}}$. Let $\mathbf{Y}_i = \hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})$. The test statistic takes on the guise

$$Q^{(n)}_{p\mathcal{G}} := (\boldsymbol{\Delta}_{\mathcal{G}}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}))'(\boldsymbol{\Gamma}_{\mathcal{G}}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}))^{-1}\boldsymbol{\Delta}_{\mathcal{G}}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}),$$

where

$$\boldsymbol{\Delta}_{\mathcal{G}}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) = n^{-1/2}\sum_{i=1}^{n}\|\mathbf{Y}_i\|\left(c_d(d+1)m_1(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})\mathbf{U}_i(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) - \|\mathbf{Y}_i\|\mathbf{S}^{\mathbf{U}}_i(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})\right)$$

and

$$\boldsymbol{\Gamma}_{\mathcal{G}}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) := \left(\frac{3}{d(d+2)}m_4(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) - 2c_d^2(d+1)m_1(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})m_3(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) + c_d^2\frac{(d+1)^2}{d}(m_1(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}))^2 m_2(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})\right)\mathbf{I}_d,$$

with $c_d = 4\Gamma(d/2)/((d^2-1)\sqrt{\pi}\Gamma(\frac{d-1}{2}))$, $\Gamma(\cdot)$ being the Gamma function. The test rejects the null hypothesis of elliptical symmetry at asymptotic level α whenever the test statistic $Q^{(n)}_{p\mathcal{G}}$ exceeds $\chi^2_{d;1-\alpha}$, the upper α-quantile of a χ^2_d distribution. We refer to Chapter 3 of Cassart (2007) for formal details.

This test can be run in our package by calling the function `pseudoGaussian()` with the simple syntax

```
pseudoGaussian(X, location = NA).
```

Besides `X`, which is a numeric matrix of data values, now we have an extra argument `location`, which allows the user to specify the known location. The default is set to `NA` which means that the unspecified location test will be performed unless the user specifies location.

SkewOptimal test

Recently, Babić et al. (2021) proposed a new test for elliptical symmetry both for specified and unspecified location. These tests are based on Le Cam's asymptotic theory of statistical experiments and are optimal against generalized skew-elliptical alternatives (defined in Section 2 of said paper), but they remain quite powerful under a much broader class of non-elliptical distributions.

The test statistic for the specified location scenario has a very simple form and an asymptotic chi-square distribution. The test rejects the null hypothesis whenever $Q^{(n)}_{\boldsymbol{\theta}} = n(\bar{\mathbf{X}} - \boldsymbol{\theta})'\hat{\boldsymbol{\Sigma}}^{-1}(\bar{\mathbf{X}} - \boldsymbol{\theta})$ exceeds the α-upper quantile $\chi^2_{d;1-\alpha}$. Here, $\hat{\boldsymbol{\Sigma}}$ is Tyler (1987)'s estimator of scatter, and $\bar{\mathbf{X}}$ is the sample mean.

When the location is not specified, Babić et al. (2021) propose tests that have a simple asymptotic chi-squared distribution under the null hypothesis of ellipticity, are affine-invariant, computationally fast, have a simple and intuitive form, only require finite moments of order 2, and offer much flexibility in the choice of the radial density f at which optimality (in the maximin sense) is achieved. Note that the Gaussian f is excluded, though, due to a singular information matrix; see Babić et al. (2021). We implemented in our package the test statistic based on the radial density f of the multivariate t distribution, multivariate power-exponential, and multivariate logistic, though in principle, any non-Gaussian choice for f is possible. The test requires lengthy notations, but its implementation is straightforward. For the sake of generality, we will derive the test statistic for a general (but fixed) f, and later on, provide the expressions of f for the three special cases implemented in our package. Let $\varphi_f(x) = -\frac{f'(x)}{f(x)}$ and $\mathbf{Y}_i = \hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_i - \hat{\boldsymbol{\theta}})$ where $\hat{\boldsymbol{\theta}}$ is the sample mean. In order to construct the test statistic, we first have to define the quantities

$$\boldsymbol{\Delta}_f(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) = 2n^{-1/2}\dot{\Pi}(0)\sum_{i=1}^{n}\left[\|\mathbf{Y}_i\| - \frac{d}{\widehat{\mathcal{K}}_{d,f}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})}\varphi_f(\|\mathbf{Y}_i\|)\right]\frac{\mathbf{Y}_i}{\|\mathbf{Y}_i\|}$$

and

$$\widehat{\boldsymbol{\Gamma}}_f(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) := \frac{4(\dot{\Pi}(0))^2}{nd}\sum_{i=1}^{n}\left[\|\mathbf{Y}_i\| - \frac{d}{\widehat{\mathcal{K}}_{d,f}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})}\varphi_f(\|\mathbf{Y}_i\|)\right]^2\mathbf{I}_d,$$

where $\widehat{\mathcal{K}}_{d,f}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}) := \frac{1}{n}\sum_{i=1}^{n}\left[\varphi'_f(\|\mathbf{Y}_i\|) + \frac{d-1}{\|\mathbf{Y}_i\|}\varphi_f(\|\mathbf{Y}_i\|)\right]$ and Π is the cdf of the standard normal distribution (we use $\dot{\Pi}(\cdot)$ for the derivative). Finally, the test statistic is of the form $Q^{(n)}_f :=$

$(\mathbf{\Delta}_f(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}))'(\widehat{\mathbf{\Gamma}}_f(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}}))^{-1}\mathbf{\Delta}_f(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\Sigma}})$, and it has a chi-square distribution with d degrees of freedom. The test is valid under the entire semiparametric hypothesis of elliptical symmetry with unspecified center and uniformly optimal against any type of generalized skew-f alternative.

From this general expression, one can readily derive the test statistics for specific choices of f. In our case, the radial density of the multivariate Student t distribution corresponds to $f(x) = (1+\frac{1}{\nu}x^2)^{-(\nu+d)^2}$, where $\nu \in (0,\infty)$ represents the degrees of freedom, while that of the multivariate logistic distribution is given by $f(x) = \dfrac{\exp(-x^2)}{[1+\exp(-x^2)]^2}$ and of the multivariate power-exponential by $f(x) = \exp\left(-\frac{1}{2}x^{2\beta}\right)$, where $\beta \in (0,\infty)$ is a parameter related to kurtosis.

These tests can be run in R using a function called `SkewOptimal()` with the syntax

```
SkewOptimal(X, location = NA, f = "t", param = NA).
```

Depending on the type of the test, some of the input arguments are not required. `X` and `location` are the only input arguments for the specified location test and have the same role as for the Pseudo-Gaussian test. As before, the default value for `location` is set to `NA`, which implies that the unspecified location test will be performed unless the user specifies the location. For the unspecified location test, besides the data matrix `X`, the input arguments are `f` and `param`. The `f` argument is a string that specifies the type of the radial density based on which the test is built. Currently supported options are `"t"`, `"logistic"`, and `"powerExp"`. Note that the default is set to `"t"`. The role of the `param` argument is as follows. If `f = "t"`, then `param` denotes the degrees of freedom of the multivariate t distribution. Given that the default radial density is `"t"`, it follows that the default value of `param` represents the degrees of freedom of the multivariate t distribution, and it is set to 4. Note also that the degrees of freedom have to be greater than 2. If `f = "powerExp"`, then `param` denotes the kurtosis parameter β, in which case the value of `param` has to be different from 1 because $\beta = 1$ corresponds to the multivariate normal distribution. The default value is set to 0.5.

Time complexity

We conclude the description of tests for elliptical symmetry by comparing their time complexity in terms of the big O notation (Cormen et al., 2009). More concretely, we are comparing the number of simple operations that are required to evaluate the test statistics and the p-values. Table 1 summarizes the time complexity of the implemented tests.

The test of Koltchinskii and Sakhanenko is computationally more demanding than the bootstrap version of the test of Huffer and Park. Among unspecified location tests that do not require bootstrap procedures, the most computationally expensive test is the MPQ test under the realistic assumption that $n > d$. Regarding the specified location tests, we can conclude that the Pseudo-Gaussian test is more computationally demanding than the SkewOptimal test. Note that both the test of Koltchinskii and Sakhanenko and the MPQ test are based on spherical harmonics up to degree 4. In case we would use spherical harmonics of higher degrees, the tests would of course become even more computationally demanding.

We have seen that several tests require bootstrap procedures and therefore are, by default, computationally demanding. Such tests require the calculation of the statistic on the resampled data R times in order to get the p-value, where R is the number of bootstrap replicates. Consequently, the time required to obtain the p-value in such cases is R times the time to calculate the test statistic. For the tests that do not involve bootstrap procedures, the p-value is calculated using the inverse of the cdf of the asymptotic distribution under the null hypothesis, which is considered as one simple operation. The exception here is the test of Huffer and Park, whose asymptotic distribution is more complicated and includes $O(c)$ operations where c is an integer and represents an input parameter for this test.

Illustrations using financial data

Mean-Variance analysis was introduced by Markowitz (1952) as a model for portfolio selection. In this model, the portfolio risk expressed through the historical volatility is minimized for a given expected return, or the expected return is maximized given the risk. The model is widely used for making portfolio decisions, primarily because it can be easily optimized using quadratic programming techniques. However, the model has some shortcomings, among which the very important one, that it does not consider the prior wealth of the investor that makes decisions. This prior wealth is important since it influences the satisfaction that an investor has from gains. For example, the gain of 50$ will not bring the same satisfaction to someone whose wealth is 1$ as to someone whose wealth is 1000$.

	statistics	p-value
KoltchinskiiSakhanenko	$O(n\log n + nd^5)$	$O(Rn\log n + Rnd^5)$
MPQ	$O(n\log n + nd^5)$	$O(1)$
Schott	$O(nd^2 + d^6)$	$O(1)$
HufferPark	$O(nd^2 + d^3)$	$O(c)$
HufferPark (bootstrap)	$O(nd^2 + d^3)$	$O(Rnd^2 + Rd^3)$
PseudoGaussian (specified location)	$O(n^2d + nd^2 + d^3)$	$O(1)$
PseudoGaussian	$O(nd^2 + d^3)$	$O(1)$
SkewOptimal (specified location)	$O(nd + d^3)$	$O(1)$
SkewOptimal	$O(nd^2 + d^3)$	$O(1)$

Table 1: Time complexity of the various tests for elliptical symmetry

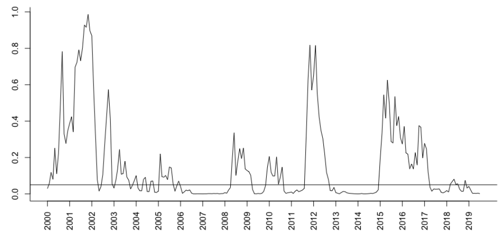
This satisfaction further affects the decision-making process in portfolio selection. Because of that and other financial reasons, a more general concept of expected utility maximization is used (see, e.g., Schoemaker (2013)). However, the expected utility maximization is not an easy optimization problem, and some additional assumptions must be made in order to solve it. Hence, despite the expected utility maximization being more general, the mean-variance approach is still used due to its computational simplicity. Chamberlain (1983) showed that the two approaches coincide if the returns are elliptically distributed. In other words, under elliptical symmetry, the mean-variance optimization solves the expected utility maximization for any increasing concave utility function. Therefore, we want to test if the assumption of elliptical symmetry holds or not for financial return data. The data set that we analyze contains daily stock log-returns of 3 major equity market indexes from North America: S&P 500 (US), TSX (Canada) and IPC (Mexico). The sample consists of 5369 observations from January 2000 through July 2020. To remove temporal dependencies by filtering, following the suggestion of Lombardi and Veredas (2009), GARCH(1,1) time series models were fitted to each series of log-returns.

We test if the returns are elliptically symmetric in different time periods using a rolling window analysis. The window has a size of one year, and it is rolled every month, i.e., we start with the window January 2000 - December 2000, and we test for elliptical symmetry. Then we shift the starting point by one month, that is, we consider February 2000 - January 2001, and we test again for elliptical symmetry. We keep doing this until the last possible window. The following tests are used for every window: the test by Koltchinskii and Sakhanenko with `R = 100` bootstrap replicates, the MPQ test, Schott's test, the bootstrap test by Huffer and Park based on orthants with `c = 3` and with the number of bootstrap replicates `R = 100`, the Pseudo-Gaussian test, and the SkewOptimal test with the default values of the parameters. For every window, we calculate the p-value. The results are presented in Figure 1, where the horizontal line present on every plot indicates the 0.05 significance level.

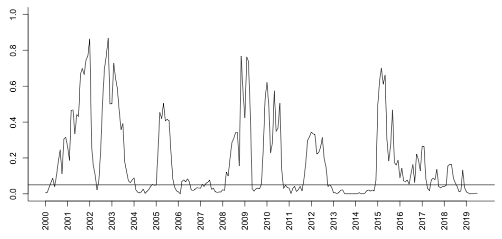
Even though all these tests address the null hypothesis of elliptical symmetry, they have different powers for different alternative distributions, and some tests may fail to detect certain departures from the null hypothesis. Certain tests are also by nature more conservative than others. We refer the reader to Babić et al. (2021) for a comparative simulation study that includes the majority of the tests available in this package. This diversity in behavior presents nice opportunities. For instance, when all tests agree, we can be pretty sure about the nature of the analyzed data. One could also combine the six tests into a multiple testing setting by using a Bonferroni correction, though this is not what we are doing here.

The following general conclusions can be drawn from Figure 1.

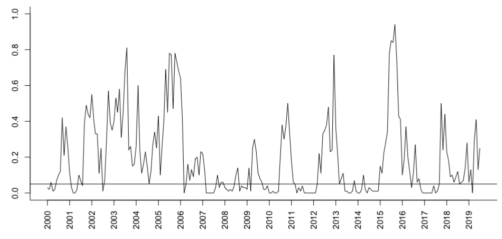
- In the past 20 years, the return data do not follow the elliptical distribution at least half of time. In other words, there are many periods between 2000 and 2020 where the data exhibit some form of skewness or other type of symmetry, invalidating, thus, the mean-variance analysis.
- The broader periods where the hypothesis of elliptical symmetry cannot be rejected are 2000-2004, 2005-2006, 2012-2013, 2015-2017 (for Schott's test only 2015-2016). In these periods, the tests may have only occasional rejections without a longer time period of rejections.
- In the period around the financial crisis in 2008, almost all tests reject the null hypothesis of ellipticity. This clearly shows that, in the periods of crisis, the assumption of elliptical symmetry is less likely to hold.



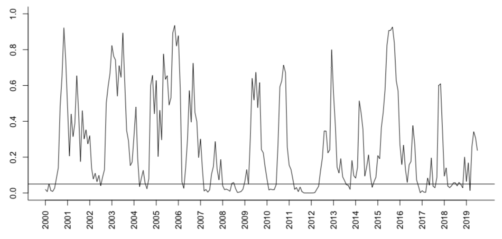
(a) SkewOptimal



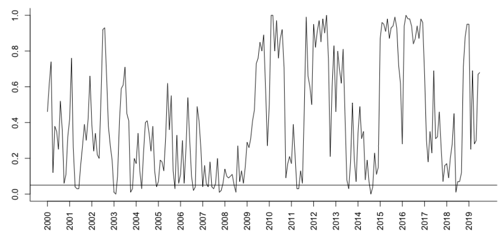
(b) Pseudo-Gaussian



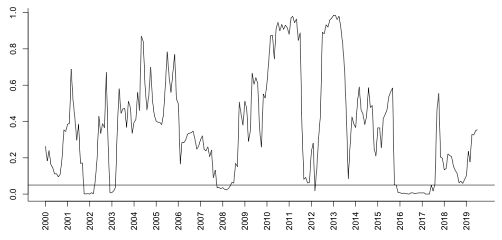
(c) KoltchinskiiSakhanenko



(d) MPQ



(e) HufferPark



(f) Schott

The plots show the p-values of the corresponding tests for all rolling windows that we considered between 2000 and 2020. The years on the x-axis mark the rolling windows for which the starting point is January of that year. The horizontal line present on every plot indicates the 0.05 significance level.

Figure 1: North America indexes (S&P, TSX and IPC)

With the aim of guiding the reader through the functions that are available in the **ellipticalsymmetry** package, we now focus on the window January 2008 - December 2008. We start with the test by Koltchinskii and Sakhanenko.

```
> KoltchinskiiSakhanenko(data2008, R = 100)

Test for elliptical symmetry by Koltchinskii and Sakhanenko

data:  data2008
statistic =  6.0884, p-value = 0.01
alternative hypothesis: the distribution is not elliptically symmetric
```

The `KoltchinskiiSakhanenko()` output is simple and clear. It reports the value of the test statistic and p-value. For this particular data set, the test statistic is equal to 6.0884 and the p-value is 0.02. Note that here we specify the number of bootstrap replicates to be `R = 100`.

The MPQ test and Schott's test can be performed by running very simple commands:

```
> MPQ(data2008)

Test for elliptical symmetry by Manzotti et al.
```

```
data:  data2008
statistic = 25.738, p-value = 0.04047
alternative hypothesis: the distribution is not elliptically symmetric

> Schott(data2008)

Schott test for elliptical symmetry

data:  data2008
statistic = 24.925, p-value = 0.03531
alternative hypothesis: the distribution is not elliptically symmetric
```

Given the number of the input arguments, the function for the test by Huffer and Park deserves some further comments. The non-bootstrap version of the test can be performed by running the command:

```
> HufferPark(data2008, c = 3)

Test for elliptical symmetry by Huffer and Park

data:  data2008
statistic = 24.168, p-value = 0.109
alternative hypothesis: the distribution is not elliptically symmetric
```

By specifying `R`, the bootstrap will be applied:

```
> HufferPark(data2008, c= 3, R = 100)
```

The p-value for the bootstrap version of the test is equal to 0.15. Note that in both cases, we used the default value for `sector`, that is `"orthants"`.

```
Test for elliptical symmetry by Huffer and Park

data:  data2008
statistic = 24.168, p-value = 0.15
alternative hypothesis: the distribution is not elliptically symmetric
```

If we want to change the type of sectors used to divide the space, we can do it by running the command:

```
HufferPark(data2008, c=3, R = 100, sector = "permutations")
```

This version yields a p-value equal to 0.19.

Another very easy-to-use test is the Pseudo-Gaussian test:

```
> PseudoGaussian(data2008)

Pseudo-Gaussian test for elliptical symmetry

data:  data2008
statistic = 9.4853, p-value = 0.02349
alternative hypothesis: the distribution is not elliptically symmetric
```

Eventually, the following simple command will run the `SkewOptimal` test based on the radial density of the multivariate t distribution with 4 degrees of freedom (note that the degrees of freedom could be readily changed by specifying the `param` argument).

```
> SkewOptimal(data2008)

 SkewOptimal test for elliptical symmetry

data:  data2008
statistic = 12.208, p-value = 0.006702
alternative hypothesis: the distribution is not elliptically symmetric
```

The test based on the radial density of the multivariate logistic distribution can be performed by simply adding `f = "logistic"`:

```
> SkewOptimal(data2008, f = "logistic")
```

This version of the SkewOptimal test yields a p-value equal to 0.0003484. Finally, if we want to run the test based on the radial density of the multivariate power-exponential distribution, we have to set `f` to `"powerExp"`. The kurtosis parameter equal to 0.5 will be used unless specified otherwise.

```
> SkewOptimal(data2008, f = "powerExp")
```

The resulting p-value equals 0.002052. The kurtosis parameter can be changed by assigning a different value to `param`. For example,

```
SkewOptimal(data2008, f = "powerExp", param = 1.2)
```

We can conclude that the null hypothesis is rejected at the 5% level by all tests except Huffer and Park's tests. Luckily the tests available in the package mostly agree. In general, in situations of discordance between two (or more) tests, a practitioner may compare the essence of the tests as described in this paper and check if, perhaps, one test is more suitable for the data at hand than the other (e.g., if assumptions are not met). The freedom of choice among several tests for elliptical symmetry is an additional feature of our new package.

Conclusion

In this paper, we have described several existing tests for elliptical symmetry and explained in detail their R implementation in our new package **ellipticalsymmetry**. The implemented functions are simple to use, and we illustrate this via a real data analysis. The availability of several tests for elliptical symmetry is clearly an appealing strength of our new package.

Acknowledgments

Slađana Babić was supported by a grant (165880) as a PhD Fellow of the Research Foundation-Flanders (FWO). Marko Palangetić was supported by the Odysseus program of the Research Foundation-Flanders.

Bibliography

S. Babić, L. Gelbgras, M. Hallin, and C. Ley. Optimal tests for elliptical symmetry: specified and unspecified location. *Bernoulli, in press*, 2021. [p661, 665, 667]

L. Baringhaus. Testing for spherical symmetry of a multivariate distribution. *The Annals of Statistics*, 19:899–917, 1991. [p661]

R. Beran. Testing for ellipsoidal symmetry of a multivariate density. *The Annals of Statistics*, 7:150–162, 1979. [p661]

D. Cassart. *Optimal tests for symmetry*. PhD thesis, Univ. libre de Bruxelles, Brussels, 2007. [p661, 664, 665]

D. Cassart, M. Hallin, and D. Paindaveine. Optimal detection of Fechner-asymmetry. *Journal of Statistical Planning and Inference*, 138:2499–2525, 2008. [p664]

G. Chamberlain. A characterization of the distributions that imply mean—variance utility functions. *Journal of Economic Theory*, 29(1):185–201, 1983. [p667]

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2009. [p666]

C. Diks and H. Tong. A test for symmetries of multivariate probability distributions. *Biometrika*, 86: 605–614, 1999. [p661]

Y. Dominicy, P. Ilmonen, and D. Veredas. Multivariate Hill estimators. *International Statistical Review*, 85(1):108–142, 2017. [p661]

R. Dyckerhoff, C. Ley, and D. Paindaveine. Depth-based runs tests for bivariate central symmetry. *Annals of the Institute of Statistical Mathematics*, 67(5):917–941, 2015. [p661]

M. Hallin and D. Paindaveine. Optimal tests for multivariate location based on interdirections and pseudo-Mahalanobis ranks. *The Annals of Statistics*, 30(4):1103–1133, 2002. [p661]

M. Hallin and D. Paindaveine. Rank-based optimal tests of the adequacy of an elliptic VARMA model. *Annals of Statistics*, 32:2642–2678, 2004. [p661]

M. Hallin and D. Paindaveine. Semiparametrically efficient rank-based inference for shape. I. Optimal rank-based tests for sphericity. *The Annals of Statistics*, 34(6):2707–2756, 2006. [p661]

M. Hallin, H. Oja, and D. Paindaveine. Semiparametrically efficient rank-based inference for shape. II. Optimal r-estimation of shape. *The Annals of Statistics*, 34(6):2757–2789, 2006. [p661]

M. Hallin, D. Paindaveine, and T. Verdebout. Optimal rank-based testing for principal components. *The Annals of Statistics*, 38(6):3245–3299, 2010. [p661]

M. Hallin, D. Paindaveine, and T. Verdebout. Efficient R-estimation of principal and common principal components. *Journal of the American Statistical Association*, 109:1071–1083, 2014. [p661]

D. J. Hodgson, O. Linton, and K. Vorkink. Testing the capital asset pricing model efficiently under elliptical symmetry: A semiparametric approach. *Journal of Applied Econometrics*, 17(6):617–639, 2002. [p661]

F. W. Huffer and C. Park. A test for elliptical symmetry. *Journal of Multivariate Analysis*, 98(2):256–281, 2007. [p661, 664]

V. Koltchinskii and L. Sakhanenko. Testing for ellipsoidal symmetry of a multivariate distribution. In *High Dimensional Probability II*, pages 493–510. Springer, 2000. [p661, 662]

R.-Z. Li, K.-T. Fang, and L.-X. Zhu. Some qq probability plots to test spherical and elliptical symmetry. *Journal of Computational and Graphical Statistics*, 6(4):435–450, 1997. [p661]

E. Liebscher. A semiparametric density estimator based on elliptical distributions. *Journal of Multivariate Analysis*, 92(1):205–225, 2005. [p661]

M. J. Lombardi and D. Veredas. Indirect estimation of elliptical stable distributions. *Computational Statistics & Data Analysis*, 53(6):2309–2324, 2009. [p667]

J. R. Magnus. Linear structures. *Griffin's statistical monographs and courses*, (42), 1988. [p663]

A. Manzotti and A. J. Quiroz. Spherical harmonics in quadratic forms for testing multivariate normality. *Test*, 10(1):87–104, 2001. [p662]

A. Manzotti, F. J. Pérez, and A. J. Quiroz. A statistic for testing the null hypothesis of elliptical symmetry. *Journal of Multivariate Analysis*, 81(2):274–285, 2002. [p661, 662]

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7:77–91, 1952. [p666]

C. Müller. *Spherical Harmonics*, volume 17 of *Lecture Notes in Mathematics*. Springer-Verlag Berlin, 1966. [p662]

J. Owen and R. Rabinovitch. On the class of elliptical distributions and their applications to the theory of portfolio choice. *Journal of Finance*, 38:745–752, 1983. [p661]

D. Paindaveine. Elliptical symmetry. In A. H. El-Shaarawi and W. Piegorsch, editors, *Encyclopedia of Environmetrics, 2nd edition*, pages 802–807. John Wiley & Sons, Chichester, UK, 2014. [p661]

L. Sakhanenko. Testing for ellipsoidal symmetry: A comparison study. *Computational Statistics & Data Analysis*, 53(2):565–581, 2008. [p661, 662]

P. J. Schoemaker. *Experiments on decisions under risk: The expected utility hypothesis*. Springer Science & Business Media, 2013. [p667]

J. R. Schott. Testing for elliptical symmetry in covariance-matrix-based analyses. *Statistics & Probability Letters*, 60(4):395–404, 2002. [p661, 663]

R. J. Serfling. Multivariate symmetry and asymmetry. In S. Kotz, N. Balakrishnan, C. B. Read, and B. Vidakovic, editors, *Encyclopedia of Statistical Sciences, Second Edition*, volume 8, pages 5338–5345. Wiley Online Library, 2006. [p661]

D. E. Tyler. A distribution-free M-estimator of multivariate scatter. *The Annals of Statistics*, 15:234–251, 1987. [p664, 665]

Y. Um and R. Randles. Nonparametric tests for the multivariate multi-sample location problem. *Statistica Sinica*, 8:801–812, 1998. [p661]

D. Vogel and R. Fried. Elliptical graphical modelling. *Biometrika*, 98(4):935–951, 2011. [p661]

L.-X. Zhu and G. Neuhaus. Nonparametric Monte Carlo tests for multivariate distributions. *Biometrika*, 87(4):919–928, 2000. [p661]

L.-X. Zhu and G. Neuhaus. Conditional tests for elliptical symmetry. *Journal of Multivariate Analysis*, 84:284–298, 2004. [p661]

Slađana Babić
Department of Applied Mathematics, Computer Science and Statistics
Ghent University
Belgium
sladana.babic@ugent.be

Christophe Ley
Department of Applied Mathematics, Computer Science and Statistics
Ghent University
Belgium
christophe.ley@ugent.be

Marko Palangetić
Department of Applied Mathematics, Computer Science and Statistics
Ghent University
Belgium
marko.palangetic@ugent.be

The vote Package: Single Transferable Vote and Other Electoral Systems in R

by Adrian E. Raftery, Hana Ševčíková and Bernard W. Silverman

Abstract We describe the **vote** package in R, which implements the plurality (or first-past-the-post), two-round runoff, score, approval, and Single Transferable Vote (STV) electoral systems, as well as methods for selecting the Condorcet winner and loser. We emphasize the STV system, which we have found to work well in practice for multi-winner elections with small electorates, such as committee and council elections, and the selection of multiple job candidates. For single-winner elections, STV is also called Instant Runoff Voting (IRV), Ranked Choice Voting (RCV), or the alternative vote (AV) system. The package also implements the STV system with equal preferences, for the first time in a software package, to our knowledge. It also implements a new variant of STV, in which a minimum number of candidates from a specified group are required to be elected. We illustrate the package with several real examples.

Introduction

The **vote** package implements several electoral methods: plurality voting, approval voting, score voting, Condorcet methods, and Single Transferable Vote (STV) methods (Ševčíková et al., 2021).

In developing the package, we were motivated particularly by the needs of organizations with small electorates, such as learned societies, clubs, and university departments, who may need to elect more than one person in a given election. In the early 1980s, one of us (BWS) was a member of the Royal Statistical Society (RSS) Council. At that time, six members of the Council were elected at a time. A nominating committee nominated six candidates, and the RSS membership as a whole voted, with each member allowed to vote for up to six candidates, and the six candidates with the most votes being elected. Usually, there were only the six nominated candidates, but that year a seventh candidate stood on a platform different from that of the "official" candidates. This candidate received votes from about a quarter of the electorate but was not elected because the other three-quarters of the members voted as a block for the six candidates proposed by the nominating committee.

This was viewed as unsatisfactory because the seventh candidate's position was not represented on the Council, even though it had substantial support among the RSS membership. This led the RSS Council to undertake a study of electoral methods for multi-winner elections, with a view to adopting a more representative system. They selected the Single Transferable Vote (STV) method, which was then adopted for Council elections, initially using a program in the Pascal programming language developed by Hill et al. (1987). In the next election, held under STV, the seventh candidate stood again and was elected. STV has been used since then to elect the RSS Council.

In 2002, the Institute of Mathematical Statistics (IMS), the leading international association of academic mathematical statisticians, considered the same issue and came to the same conclusion, also adopting STV for its Council elections. They used an R program developed by BWS (Silverman, 2002, 2003), who was also then the IMS President. This R program became the core of the **vote** package that we are describing here. This STV electoral method has been used since then by the IMS.

Since then, another one of us (AER) has implemented the STV method in the context of small electorates selecting or ranking multiple candidates, such as nominating committees selecting multiple awardees for a prize, or academic departments selecting job candidates for interviews. Those involved have generally reported finding the method satisfactory. This experience has led to several modifications of the program that are also implemented in the package.

Our implementation and discussion of STV and other systems is aimed particularly at those involved in non-party-political elections and decisions, such as those outlined above. Questions of what approaches are or are not desirable for national elections are matters of political science beyond the scope of this paper, which is not intended to advocate for or against the use of any particular voting systems in that context. However, a brief review may be informative.

The USA and the UK, for their national legislatures, almost entirely use the plurality, "first past the post" or "winner takes all" system, where the leading candidate in each district is elected. The Electoral College for the US presidency is also elected this way, but with an election between slates rather than individuals, in all states except Maine and Nebraska. On the other hand, the majority of countries use some system that (in principle at least) aims for the elected body to represent the views of the wider electorate proportionately, either over the country as a whole or within larger electoral districts. However, pure proportional systems are fairly unusual, for example because in nationwide proportional systems there is often a threshold below which a party will not have any representation. The Single Transferable Vote system is used to elect the parliaments or national assemblies of the Republic of Ireland, Northern Ireland, and Malta, as well as upper houses and/or local assemblies in some other countries (Wikipedia, 2020c), and we draw an example from a Dublin election in the paper.

As we have said, it is not our purpose to advocate any one electoral method, and indeed it is well known that there is no one method that dominates all others given a reasonable set of criteria, according to the impossibility theorems of Arrow (1963), Gibbard (1973) and Satterthwaite (1975). Indeed, method choice can depend on the purpose of the election, and a method that works well for one purpose (such as representing the views of the electorate), may not be best for others (such as electing an effective team to work together) (Syddique, 1988). As a result, we have implemented multiple electoral methods in the package. Pros and cons of a wide range of different electoral systems are described in Ace Project (2020), but these focus on nationwide political elections, whereas here we also pay attention to smaller, often non-political elections, such as those for councils and committees.

The paper is organized as follows. In Section 2.2, we describe the plurality, two-round runoff, approval, score, and Condorcet vote-counting methods. In Section 2.3, we describe the STV method, including the first software implementation of the equal preference STV method, to our knowledge. This also describes a new variant of STV, which enforces minimal representation of a marked group. In Section 2.4, we describe three multi-winner elections with electorates of different sizes: an election from one constituency in the 2002 Irish General Election, an election of the IMS Council, and a vote to select job candidates by a university department. We conclude in Section 2.5 with the discussion of issues including other R packages for vote-counting.

Electoral methods

In this section, we describe several electoral methods and how they are implemented in the **vote** package. We defer description of STV to Section 2.3.

We first illustrate the results here with the toy `food_election` dataset:

```
> library (vote)
> data (food_election)
> food_election

   Oranges Pears Chocolate Strawberries Sweets
1       NA    NA         1            2     NA
2       NA    NA         1            2     NA
3       NA    NA         1            2     NA
4        2     1        NA           NA     NA
5       NA    NA        NA            1     NA
6        1    NA        NA           NA     NA
7       NA    NA        NA           NA      1
8        1    NA        NA           NA     NA
9       NA    NA         1            2     NA
10      NA    NA         1           NA      2
11       1    NA        NA           NA     NA
12      NA    NA         1            2     NA
13      NA    NA         1            2     NA
14      NA    NA         1            2     NA
15      NA    NA         1           NA      2
16       1    NA        NA           NA     NA
17      NA    NA         1           NA      2
18       2     1        NA           NA     NA
19      NA    NA         1           NA      2
20      NA    NA         1            2     NA
```

In this toy dataset, voters were asked to rank the options in order of preference. They gave only their first two preferences, although they could have given more; an `NA` indicates that no preference was expressed.

Plurality voting

Plurality voting, or First-Past-The-Post, is used for single-winner elections, such as elections to the House of Representatives in the USA or the House of Commons in the UK. Each voter votes for one candidate, and the candidate with the most votes wins.

To implement this with our toy dataset, we first converted it to a dataset where only first preferences count:

```
> food_election_plurality <- 1 * (food_election == 1 & !is.na (food_election))
> head(food_election_plurality)

     Oranges Pears Chocolate Strawberries Sweets
[1,]       0     0         1            0      0
[2,]       0     0         1            0      0
[3,]       0     0         1            0      0
[4,]       0     1         0            0      0
[5,]       0     0         0            1      0
[6,]       1     0         0            0      0
```

We then counted the votes using the `plurality` command:

```
> plurality (food_election_plurality)

Results of Plurality voting
===========================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          1


|    |Candidate    | Total| Elected |
|:---|:------------|-----:|:-------:|
|1   |Chocolate    |    12|    x    |
|2   |Oranges      |     4|         |
|3   |Pears        |     2|         |
|4   |Strawberries |     1|         |
|5   |Sweets       |     1|         |
|Sum |             |    20|         |

Elected: Chocolate
```

Plurality voting has the advantage of simplicity. In political elections, it tends not to yield results that are in direct proportion to support among the voters but to amplify pluralities when compared to proportional voting systems, which merge single-winner districts into larger multi-member groups. In general, any large party which has strong support across a large number of electoral districts will do well under plurality voting, while smaller parties or interests will tend to be underrepresented numerically, especially if they are evenly or thinly spread. This may mean that important interests are not represented, while on the other hand, it may present a barrier to the traction of extremist groups. The US Electoral College is, in nearly all states, elected by a plurality voting system, with multiple members, all being elected simultaneously.

Plurality voting in individual-member districts tends to lead to one-party governments with working majorities, even when the leading party does not achieve half of the popular vote. It also allows districts to be smaller to facilitate direct contact between a voter and their representative and identifies each representative more closely with all the voters in their district.

Another effect of plurality voting can be to "waste" the votes of those who live in highly polarised districts, because they win their particular district by a very wide margin; this seems to be a deliberate feature of much redistricting in the USA. In non-political elections in the smaller contexts of primary concern in this paper, there is little or no need for a stable one-"party" result, and the desirability of closer proportional representation of the views of the voters is less contentious, and so there is likely to be a clearer case for using other voting systems wherever possible.

Two-round runoff voting

Two-round systems are also used for single-winner elections. In the first round, voters vote for their first preference. If no candidate gets a majority, there is a second round that involves the top two candidates. Voters vote again, and the candidate getting more votes wins.

In the **vote** package, we implemented a variant of this system that can be counted in a single pass over the data. Each voter ranks the candidates in order of preference. The first round takes place as described. The second round is counted as if voters voted for the remaining candidate for which they had a higher preference.

To illustrate the two-round runoff system, we modify the food election data by removing voters 12–15, so that Chocolate does not have a majority on the first round:

```
> food_election3 <- food_election[-c(12:15),]
> tworound.runoff (food_election3)

Results of two-round-runoff voting
==================================
Number of valid votes:   16
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          1


|    |Candidate    | Total| Percent| ROffTotal| ROffPercent| Elected |
|:---|:------------|-----:|-------:|---------:|-----------:|:-------:|
|1   |Oranges      |     4|    25.0|         6|        42.9|         |
|2   |Pears        |     2|    12.5|         0|         0.0|         |
|3   |Chocolate    |     8|    50.0|         8|        57.1|    x    |
|4   |Strawberries |     1|     6.2|         0|         0.0|         |
|5   |Sweets       |     1|     6.2|         0|         0.0|         |
|Sum |             |    16|   100.0|        14|       100.0|         |

Elected: Chocolate
```

We see that no candidate got a majority on the first round, although Chocolate came close. In the second round, the two top vote-getters, Chocolate and Oranges, squared off, and Chocolate won.

In the `tworound.runoff` function, a tie in either the first or the runoff round is resolved by random draw. A random seed can be specified so that the results are replicable.

Two-round elections are quite common, most famously for French presidential elections since 1965. In practice, it is usually carried out by voters actually voting twice, rather than ranking candidates as here. An exception to it is a special case of the two-round runoff, called supplementary voting, where voters give only their first and second preferences on one ballot, the same way as happened in our food example. Supplementary voting is used, for example, in electing mayors in England, including the Mayor of London (London Elects, 2020).

The two-round runoff system differs from plurality voting in that voters for candidates with low levels of support can change their votes to one of the leading candidates so that they can express support for a possibly less popular first choice without their vote being "wasted". Of course, the choice between the two finalists shares some of the aspects of plurality voting.

Approval voting

Approval voting was advocated by Brams and Fishburn (1978). In this system, voters vote for as many candidates as they wish. It has been most often advocated for single-winner elections, in which case the winner is the candidate with the most votes (Brams and Fishburn, 2007). A direct extension to multi-winner elections with m winners is that voters vote in the same way, and the m candidates with the most votes win.

Counting the votes is simple. The argument `nseats` determines the number of winners m:

```
> food_election_approval <- 1 * !is.na (food_election)
> approval (food_election_approval, nseats = 2)

Results of Approval voting
==========================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2


|    |Candidate    | Total| Elected |
|:---|:------------|-----:|:-------:|
|1   |Chocolate    |    12|    x    |
|2   |Strawberries |     9|    x    |
|3   |Oranges      |     6|         |
|4   |Sweets       |     5|         |
|5   |Pears        |     2|         |
|Sum |             |    34|         |

Elected: Chocolate, Strawberries
```

Approval voting for multi-winner elections has been criticized on various grounds, e.g., Hill (1988), and indeed in the book by Brams and Fishburn (1983) that advocated and popularized approval voting for single-winner elections. For elections in which there are parties or slates of candidates, it would tend to lead to the election of all the members of the most supported party or slate, as happened in the RSS Council election that first motivated this work. However, one of us [AER] has participated in multi-winner elections using approval voting and has observed it to work well, particularly when there are many candidates about whom information is limited and there are no parties or slates. One example could be the early stages of job candidate selection when a long list is being whittled down to a small set of finalists.

Score voting

In the score or range voting, each voter gives each candidate a score within a prespecified range. If the voter does not give a score to a particular candidate, a corresponding prespecified score is assigned. The candidates with the lowest scores win (or the highest scores if higher scores are better). In the `score` function, the argument `larger.wins` specifies whether lower scores are better or higher scores are better. The argument `max.score` sets the prespecified non-vote score. Here, we illustrate score voting by applying it to the food election example, where the score is equal to the preference, a non-vote is assigned a value of 6, and lower scores are better:

```
> score (food_election, larger.wins = FALSE, nseats = 2, max.score = 6)

Results of Score voting
=======================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2


|    |Candidate    | Total| Elected |
|:---|:------------|-----:|:-------:|
|1   |Chocolate    |    60|    x    |
|2   |Strawberries |    83|    x    |
|3   |Oranges      |    92|         |
|4   |Sweets       |    99|         |
|5   |Pears        |   110|         |
|Sum |             |   444|         |

Elected: Chocolate, Strawberries
```

Score voting is often used by committees for purposes such as selecting grant applications to be funded. In such cases, there are often many candidates or applications to be assessed, and it would not be feasible for the voters to produce a complete ranking. Score voting is similar to multi-winner approval voting in this sense but allows for a more refined assessment by the voters. Multi-winner approval voting is actually a special case of score voting.

Condorcet method

The Condorcet method is attributed to Marquis de Condorcet (de Condorcet, 1785). It is a single-winner method where voters rank the candidates according to their preferences. The so-called *Condorcet winner* is the candidate who wins the majority of votes in all head-to-head comparisons. In other words, each candidate is compared pairwise to all other candidates. To become the Condorcet winner, one has to win all such comparisons. Analogously, a *Condorcet loser* is the candidate who loses in every pairwise comparison.

The condorcet function can be applied directly to the food election data:

```
> condorcet(food_election)

Results of Condorcet voting
===========================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          1


|             | Oranges| Pears| Chocolate| Strawberries| Sweets| Total| Winner | Loser |
|:------------|-------:|-----:|---------:|------------:|------:|-----:|:------:|:-----:|
|Oranges      |       0|     1|         0|            0|      1|     2|        |       |
|Pears        |       0|     0|         0|            0|      0|     0|        |   x   |
|Chocolate    |       1|     1|         0|            1|      1|     4|   x    |       |
|Strawberries |       1|     1|         0|            0|      1|     3|        |       |
|Sweets       |       0|     1|         0|            0|      0|     1|        |       |

Condorcet winner: Chocolate
Condorcet loser: Pears
```

The output above shows the results of all the pairwise comparisons. Chocolate beat all other candidates and was, therefore, the Condorcet winner. Similarly, Pears lost against all other candidates and was thus the Condorcet loser.

The Condorcet method does not guarantee that a Condorcet winner exists. There are many different ways to deal with such a situation; see, for example, Wikipedia (2020a). Our implementation offers the possibility of a runoff (argument runoff). In this case, two or more candidates with the most pairwise wins are selected, and the Condorcet method is applied to such subset. If more than two candidates are in such a runoff, the selection is performed repeatedly until either a winner is selected or no more selection is possible.

To our knowledge, the Condorcet method is not used for governmental elections anywhere in the world. Wikipedia (2020a) cites a few private organizations that use the method, e.g., the Student Society of the University of British Columbia.

Single Transferable Vote (STV)

The Single Transferable Vote (STV) system is also referred to as Ranked Choice Voting (RCV), Instant Runoff Voting (IRV), or the Alternative Vote (AV) system for single-winner elections, and as Multi-Winner Ranked Choice Voting for multi-winner elections. One of the properties of the Single Transferable Vote system is that if any subset of candidates gets a sufficient share of the votes, anything strictly exceeding $1/(m+1)$, where m is the number of candidates to be elected, then one of this group is bound to be elected. To be precise, what is required is that a proportion above $1/(m+1)$ of the voters have to put all the candidates in the subset at the top of their list of preferences, but it does not matter in what order. This would apply equally if the subset was a particular slate/party, or specified by some other group characteristic such as sex, race, geographical location, career stage, or subject area, even if the subset was not consciously constituted. In particular, if a candidate's proportion of the first preference votes is above $1/(m+1)$, then that candidate will be successful.

There is also the fact that a group is not disadvantaged if more of its members stand for election, at least if their voters vote along group lines the full way down the preferences. Unlike in some other systems, they cannot cancel each other out.

When STV was adopted for the elections of the Council of the RSS in the mid-1980s and the IMS in 2002, it was hoped that it would lead to more diverse Councils than the results of the previous methods, and also that individual members, other than those chosen by the nominating committee, would feel able to stand with a real chance of being elected.

STV method

There are many descriptions of the STV system (Newland et al., 1997; Fair Vote, 2020) and its history (Hill, 1988; Tideman, 1995). The basic principle is that voters rank the candidates in order of preference. In order to be elected, a candidate must achieve the quota of $N/(m+1)+\varepsilon$, where N is the total number of votes cast, m is the number of candidates to be elected (or seats), and ε is a pre-specified small positive number, often taken to be 1 when the electorate is large and 0.001 when it is small. Excess votes over the quota are appropriately downweighted and allocated to the next preference of voters. If no candidate reaches the quota, the candidate with the smallest number of votes is eliminated and his or her votes are transferred to the next preferences.

Voters are asked to rank the candidates $1, 2, 3, \ldots$ until they have no further preference between candidates. Thus 1 is a voter's first preference, 2 is their next choice, and so on. There is no disadvantage to higher candidates in expressing a full list of preferences; later preferences are used only when the fate of candidates given higher preferences has been decided one way or the other.

By default, a vote is considered spoiled if the preferences are not numbered consecutively, starting at 1. However, if this is not desired, the votes can be preprocessed to be consecutive using the `correct.ranking` function in the **vote** package. A useful application of this correction is the case when a candidate has to be removed, perhaps because of having withdrawn his or her candidacy. In this case, the function `remove.candidate` can be used, which removes the given candidate(s) from the set of votes, and also adjusts the preferences to be consecutive. The package optionally allows the user to accept a partially correct ranking. That is, only preferences equal to or higher than the non-consecutive rankings are removed. For example, with this option, a valid version of a vote $1, 2, 3, 4, 4, 5$ would be $1, 2, 3, 0, 0, 0$.

Also by default, apart from the candidates not numbered at all, no ties are allowed among the numbered preferences. However, equal preferences can be allowed by using the setting `equal.ranking=TRUE` in the `stv` function, as described in more detail in Section 2.3.4.

The fact that some voters may not express a full list of preferences can be allowed for by reducing the quota in later counts[1]. In the **vote** package, the default is that the quota is reduced in later counts. However, in some STV systems (such as the electoral system in the Republic of Ireland), the quota remains constant over counts at the value that is initially defined. This is specified in the **vote** package by setting the argument `constant.quota=TRUE` in the `stv` function. In this implementation of STV, the last candidate is often elected without reaching the quota, which does not happen when the quota is reduced appropriately at each count.

In the **vote** package, the votes should be entered into a matrix or data frame, with the header containing the names of the candidates and each row the votes cast, with blank preferences being replaced by zeroes or `NA`s. This will often be done by entering the votes into a spreadsheet first and then reading the spreadsheet into R. If the data are stored in a text file, the package allows one to pass the name of the file directly into the `stv` function while setting the column separator in the `fsep` argument.

At the end of the process, the program yields a list of the successful candidates in the order in which they were elected. It also usually yields a complete ordering of the candidates. This may be useful, for example, if the purpose of the election is to select job candidates, and one wishes to have an ordered list of the initially unsuccessful candidates in case any of those selected decline the offer. Also, in some systems, candidates can claim expenses if a certain rank is achieved, which could be another motivation for a runner-up list being available.

Until the 1980s, STV elections were counted manually by physically transferring a sample of the ballot papers from the pile of the candidate being elected or eliminated, to those of the benefitting candidates. This remains the case in several long-established STV election systems, such as elections for the Dáil (the lower house of the Irish parliament). Meek (1969, 1970) described the form a computer-based STV system could take, and this was implemented in Pascal by Hill et al. (1987). This code was used for the RSS Council elections. A modified version was implemented in R by Silverman (2002, 2003), and this was the starting point for the current STV implementation in the **vote** package.

Here is the result of the food election with two candidates to be elected, using the `stv` function:

```
> stv (food_election, nseats = 2)

Results of Single transferable vote
===================================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2
```

[1]In STV, the process of distributing the surplus or votes of a candidate who is elected or eliminated is referred to variously as a count, a stage, or a round. Here, we use the term count. The tabulation of the first preference votes is then called the first count.

```
|            |         1| 2-trans|     2| 3-trans|      3| 4-trans|       4|
|:-----------|---------:|-------:|-----:|-------:|------:|-------:|-------:|
|Quota       |     6.668|        | 6.667|        |  6.667|        |   5.278|
|Oranges     |     4.000|   0.000| 4.000|       2|  6.000|   0.000|   6.000|
|Pears       |     2.000|   0.000| 2.000|      -2|       |        |        |
|Chocolate   |    12.000|  -5.332|      |        |       |        |        |
|Strawberries |     1.000|   3.555| 4.555|       0|  4.555|   0.000|   4.555|
|Sweets      |     1.000|   1.777| 2.777|       0|  2.777|  -2.777|        |
|Elected     | Chocolate|        |      |        |       |        | Oranges|
|Eliminated  |          |        | Pears|        | Sweets|        |        |

Elected: Chocolate, Oranges
```

Oranges was elected second, whereas, under the approval vote system with first and second preferences treated equally, Strawberries was elected second. This reflects the fact that Oranges had 4 first preferences, whereas Strawberries had only 1. Under STV, a vote is credited entirely to the first preference candidate unless that candidate is elected or eliminated, in which case the second preferences come into play. Strawberries had 8 second-preference votes, all of which were from voters who voted for Chocolate first. The quota was only 56% of the votes for Chocolate, and so 44% of Chocolate's votes were transferred when Chocolate was elected. Strawberries gained 3.555 votes this way from its second preference votes, but this was not quite enough to overcome Orange's advantage in first preferences. The complete ordering of candidates can be read off the results: Chocolate, Oranges, Strawberries, Sweets, Pears. Setting the argument `complete.ranking` to `TRUE` will include the complete ordering as part of the output.

The package has several functions for visualizing the STV results, and we will illustrate these in the Examples section below. In addition, `summary` functions are available for the resulting objects of all voting methods in the package. In the case of `stv`, the `summary` function returns a data frame containing the table shown in the above output, which can be used for further processing, for example, for storing in a spreadsheet.

Computational methods

The algorithm used for counting STV elections using the `stv` function in the **vote** package is shown in Algorithm 1. There are only two changes needed to implement STV with equal preferences; these are shown in Section 2.3.4.

Tie-breaking

Suppose that on a given count, no candidate is elected, and a candidate needs to be selected for elimination, and that two or more candidates are tied with the smallest number of votes. Then a method is needed for choosing the one to be eliminated. The same issue arises when two candidates can be elected on the same count with the same number of votes, namely which surplus to transfer first.

Several different methods have been proposed. The Electoral Reform Society, one of the leading organizations advocating the use of STV, recommends using the Forwards Tie-Breaking Method (Newland et al., 1997, Section 5.2.5). Other methods such as Backwards Tie-Breaking, Borda Tie-Breaking, Coombs Tie-Breaking, or a combination of those have been proposed; see, e.g., O'Neill (2004); Kitchener (2005); Lundell (2006).

By default, the **vote** package uses the Forwards Tie-Breaking Method. This consists of eliminating/electing the candidate who had the fewest/most votes on the first count or on the earliest count where they had unequal votes. If the argument `ties` in the `stv` function is set to "b", the Backwards Tie-Breaking Method is used. In this case, it eliminates/elects the candidate who has the fewest/most votes on the latest count where the tied candidates had unequal votes.

There is no guarantee that a tie will be broken by either the Forwards or Backwards Tie-Breaking Method. Also, if one of these two methods does not break the tie, the other will not either because the tied candidates will have the same number of votes in all the counts so far. In particular, this will be the case whenever a tie has to be broken on the first count, and it is also relatively likely when a tie arises on the second count.

When there is a tie that Forwards and Backwards Tie-Breaking fail to break, the `stv` function uses a method that compares the candidates on the basis of the numbers of individual preferences. We call this the *Ordered* method as it creates an ordering of the candidates before the STV count begins. First, candidates are ordered by the number of first preferences. Any ties are resolved by proceeding to the

```
procedure STV($X, m, \varepsilon$)
        ▷ $X$ are votes of size $N$ (number of votes) $\times M$ (number of candidates)
    $D \leftarrow \{1, 2, \dots, M\}$    ▷ Set of *hopeful* candidates
    $E \leftarrow \{\}$    ▷ Set of *elected* candidates
    $F \leftarrow \{\}$    ▷ Set of *eliminated* candidates
    $L \leftarrow m$    ▷ Remaining number of seats
    $Y \leftarrow X$    ▷ Remaining votes
    $c \leftarrow 0$    ▷ Which Count we are at
    $w_i \leftarrow 1 \quad \forall i = 1, \dots, N$    ▷ Initialize a vector of weights, one per voter
    while $L > 0$ do    ▷ End if there are no remaining seats
        $c \leftarrow c + 1$    ▷ Increase Count
        $u_{i,j} \leftarrow w_i \delta_{Y_{i,j}}(1) \quad \forall i = 1, \dots, N,\ j = 1, \dots, M$    ▷ Weighted first preferences
        $v_{c,j} \leftarrow \sum_{i=1}^{N} u_{i,j} \quad \forall j = 1, \dots, M$    ▷ Sum of weighted first preferences
        $Q \leftarrow \sum_{j=1}^{M} v_{c,j} / (L + 1) + \varepsilon$    ▷ Compute quota
        if $\max_{j \in D} v_{c,j} \geq Q$ then    ▷ A candidate is to be elected
            $k \leftarrow \arg\max_{j \in D} v_{c,j}$    ▷ Which candidate has the most votes
            if $||k|| > 1$ then    ▷ If there is more than one such candidate
                $k \leftarrow$ resolve.tie.for.election($k, X, v$)    ▷ Break tie
            end if
            $S \leftarrow (\max_{j \in D} v_{c,j} - Q) / \max_{j \in D} v_{c,j}$    ▷ Compute surplus
            $w_r \leftarrow u_{rk} * S \quad \forall r$ where $Y_{r,k} = 1$    ▷ Recompute weights
            $L \leftarrow L - 1$    ▷ Decrease number of available seats
            $E \leftarrow E \cup \{k\}$    ▷ Candidate $k$ is elected
        else    ▷ A candidate is to be eliminated
            $k \leftarrow \arg\min_{j \in D} v_{c,j}$    ▷ Which candidate has the least votes
            if $||k|| > 1$ then    ▷ If there is more than one such candidate
                $k \leftarrow$ resolve.tie.for.elimination($k, X, v$)    ▷ Break tie
            end if
            $F \leftarrow F \cup \{k\}$    ▷ Candidate $k$ is eliminated
        end if
        $D \leftarrow D \backslash \{k\}$    ▷ Candidate $k$ is removed from the pool of hopefuls
        $Y_{i,r} \leftarrow Y_{i,r} - 1 \quad \forall i = 1, \dots, N$ where $Y_{i,k} > 0$ and $r = 1, \dots, M$ where $Y_{i,r} > Y_{i,k}$
                ▷ Above: shift votes for voters who voted for candidate $k$
        $Y_{i,k} \leftarrow 0 \quad \forall i = 1, \dots, N$    ▷ Remove votes for candidate $k$
    end while
    return($E, F, v$)
end procedure
```

Note: $\delta_Y(1) = 1$ if $Y = 1$ and 0 otherwise, is the Kronecker delta function; the arg max and arg min functions return sets, with more than one element when there is a tie; and $||k||$ is the number of elements in the set k.

Algorithm 1: STV algorithm. The input data consist of a matrix X of the votes of size $N \times M$, with N being the number of ballots and M the number of candidates. m is the number of seats to be filled and ε is a small number used for defining the quota.

total number of second preferences, then the third preferences, and so on. If a tie cannot be resolved even by counting the last preference, then it is broken by a random draw with equal probabilities for the tied candidates. A random seed is specified so that the result is replicable.

Combining Forwards and Backwards Tie-Breaking with the Ordered method and random sampling, each tie in the `stv` function is broken in one of the following three ways:

1. Forwards ("f") or Backwards ("b") Tie-Breaking method alone
2. Forwards or Backwards Tie-Breaking followed by the Ordered method ("fo", "bo")
3. Forwards or Backwards Tie-Breaking followed by the Ordered method, and finally, random sampling ("fos", "bos")

The abbreviation of these three possibilities in parentheses is included in the STV output whenever a tie is broken during the election count.

Ties of any kind are relatively rare unless the electorate is small. In very small electorates, ties are more common, but cases where Forwards, Backwards, and Ordered Tie-Breaking all fail to break the tie are unusual even then, so election by random draw will be a rare event.

In the earliest version of the software (Silverman, 2002), ties were broken deterministically: if a candidate was to be elected, the last-named member of a tie was chosen. On the other hand, if there was a tie for elimination, it was the first-named who was eliminated. These choices were aimed at compensating in a small way for the tendency of candidates higher up the ballot paper to get more votes. However, they depended on position on the ballot paper, which might be viewed as somewhat arbitrary, and in the **vote** package, we have used a more systematic criterion.

Equal preference STV

Extant implementations of STV require that voters not give equal preferences (except among the candidates that they do not rank). However, Meek (1970) has pointed out that the single transferable vote system does not exclude this possibility and outlined how the votes might be counted. This has never been implemented before in software, to our knowledge, although it is used for the election of the Trustees of the John Muir Trust Wikipedia (2020c).

The basic idea is that if, for example, a voter gives their first preference to candidates A and B, then the vote will be equally split between the two, giving half a vote to each. If A is elected, then the proportion of the half-vote for A corresponding to A's surplus will be transferred to their next highest preference. This will be B if B is still in contention, i.e., if B has not been elected or eliminated by that stage. Otherwise, it will be the remaining candidate with the next highest preference from that voter. Similarly, if A is eliminated, the half-vote for A will be fully transferred to their next highest remaining preference. This will be B if B is still in contention, or otherwise the candidate with the next highest preference.

The same principle applies if there are three or more equal preferences. For example, consider the case where there are three equal preferences A, B, and C, and A is eliminated/elected. If A is elected, the proportion of the one-third vote for A corresponding to A's surplus is equally divided between B and C. If A is eliminated, then both B and C get increased to a half vote. Algebraically, this is implemented by the change below in Line 11 of Algorithm 1.

Otherwise, the count proceeds in the same way as when equal preferences are not allowed. The argument `equal.ranking` in the `stv` function is set to `TRUE` when equal preferences are allowed. In this case, votes are postprocessed before counting so that they correctly reflect preferences. For example, a vote 1, 1, 2, 3, 3, 3 would be recoded to 1, 1, 3, 4, 4, 4. This is in contrast with the usual case where equal preferences are not allowed and `equal.ranking=FALSE` when votes with non-sequential preferences, such as 1, 2, 4, 5, are declared invalid and considered spoiled unless a partial correction is allowed.

STV with equal preferences can be implemented by Algorithm 1 with only two relatively small changes, namely:

Line 11 replaced by: $u_{i,j} \leftarrow w_i \delta_{Y_{i,j}}(1) / \sum_{\ell=1}^{M} \delta_{Y_{i,\ell}}(1) \quad \forall\, i = 1, \ldots, N,\ j = 1, \ldots, M$

Line 20 replaced by: $w_r \leftarrow \sum_{j=1}^{M} u_{r,j} - u_{r,k} + u_{r,k} * S \quad \forall\, r$ where $Y_{r,k} = 1$

Note that if applied to votes with no equal preferences, the modified algorithm yields the same result as Algorithm 1. In such a case, the denominator in Line 11 is equal to 1 for all i, and thus, u is the same as in Algorithm 1. Similarly, in Line 20, $\sum_{j=1}^{M} u_{r,j} - u_{r,k} = 0$ if there are no equal preferences, and thus, w is the same as in Algorithm 1.

We illustrate this functionality using the food election data by setting the first three votes to equal first preferences for Chocolate and Strawberries, instead of first and second preferences:

```
> food_election2 <- food_election
> food_election2[c(1:3), 4] <- 1
> stv (food_election2, equal.ranking = TRUE)

Results of Single transferable vote with equal preferences
============================================================
Number of valid votes:   20
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2


|             |         1| 2-trans|     2| 3-trans|      3| 4-trans|       4|
|:------------|---------:|-------:|-----:|-------:|------:|-------:|-------:|
|Quota        |     6.668|        | 6.667|        |  6.667|        |   5.437|
|Oranges      |     4.000|   0.000| 4.000|       2|  6.000|   0.000|   6.000|
|Pears        |     2.000|   0.000| 2.000|      -2|       |        |        |
|Chocolate    |    10.500|  -3.832|      |        |       |        |        |
|Strawberries |     2.500|   2.372| 4.872|       0|  4.872|   0.000|   4.872|
|Sweets       |     1.000|   1.460| 2.460|       0|  2.460|  -2.460|        |
|Elected      | Chocolate|        |      |        |       |        | Oranges|
|Eliminated   |          |        | Pears|        | Sweets|        |        |

Elected: Chocolate, Oranges
```

Once again, Oranges is elected second, ahead of Strawberries, although the margin of victory is smaller than before.

Reserved seats in STV

In addition to having a given number of seats to fill, it may be desired to elect a minimum number of candidates from a specified class or group of candidates. For example, the selectors of plenary papers at a conference might wish to reserve at least two slots for students. Or the election of a committee might wish to ensure that at least three women were elected.

We have incorporated this feature into the `stv` function as an option. Users can specify the number of reserved seats with the argument `group.nseats` and mark the members eligible for those seats in the argument `group.members`.

When this requirement is present, we modify our STV algorithm as follows. Suppose m denotes the number of seats and g denotes the number of reserved seats, and candidates are either *marked* (eligible for reserved seats) or *unmarked* (not eligible). Then on each count,

- if the leading candidate exceeds the quota, they are elected, except that if $m - g$ unmarked candidates have already been elected, they are only elected if they are marked. Or,
- if no candidate has been elected on this round, the candidate with the fewest votes is eliminated, except that if there are only g marked candidates still in play (including any already elected) or if there are already $m - g$ unmarked candidates elected, the unmarked candidate with the fewest votes is eliminated (even if that number of votes is above the quota).

We will illustrate the reserved seats feature in Section 2.4.3.

We are not aware of any previous method in the literature to allow for reserved seats in STV. It seems possible that other methods for doing this could be developed. For example, one could just eliminate all remaining unmarked candidates once $m - g$ unmarked candidates had been elected. It is not clear if or when this would give different results.

In principle, this general approach could be extended to the situation where there are two or more constraints, such as ensuring that at least three women and at least two students are elected to a committee of 12. The basic principles are the same:

- When eliminating a candidate, make sure that there are enough of the right kinds of candidates still in play that all remaining constraints can be satisfied.
- When choosing a candidate to elect, make sure there are enough slots left so that the remaining constraints can be satisfied later.

However, in the **vote** package, for now, it is implemented only for one constraint.

Examples

We now illustrate the different systems using three examples of elections. Perhaps ironically, systems are more robust with larger than smaller electorates, in the sense that their results are less sensitive to small changes in the electoral system. We therefore start with a political election with a relatively large electorate, continue with the election of the council of a scientific organization with a moderate-sized electorate, and finally describe an election with a very small electorate. Each of these was a multi-winner election, but we will also use them to illustrate the single-winner electoral methods.

Irish general election 2002: Dublin West constituency

The Dublin West constituency in the 2002 Irish general election had three seats to be filled, nine candidates, and just under 30,000 ranked votes. The dataset, called `dublin_west`, is included in the package. These data were collected electronically in a trial of electronic voting, with a system that prevented invalid votes. As a result, there were no invalid votes, unusually for an election of this kind.

```
> data (dublin_west)
> head(dublin_west)

  Bonnie Burton Ryan Higgins Lenihan McDonald Morrissey Smyth Terry
1      0      4    0       3       0        0         1     5     2
2      0      0    2       0       1        4         3     0     0
3      0      0    3       0       1        0         2     0     0
4      0      2    0       0       0        0         3     0     1
5      0      2    1       0       0        0         0     0     0
6      0      3    2       0       1        0         0     0     0
```

We illustrate the single-winner methods by assuming that there is just one seat to be filled. First we consider the plurality method. It is necessary to convert the dataset into a set of zeros and ones to run the `plurality` function:

```
> dublin_west1 <- 1*(dublin_west == 1)
> plurality (dublin_west1)

Results of Plurality voting
===========================
Number of valid votes:   29988
Number of invalid votes:     0
Number of candidates:        9
Number of seats:             1


|    |Candidate | Total| Elected |
|:---|:---------|-----:|:-------:|
|1   |Lenihan   |  8086|    x    |
|2   |Higgins   |  6442|         |
|3   |Burton    |  3810|         |
|4   |Terry     |  3694|         |
|5   |McDonald  |  2404|         |
|6   |Morrissey |  2370|         |
|7   |Ryan      |  2300|         |
|8   |Bonnie    |   748|         |
|9   |Smyth     |   134|         |
|Sum |          | 29988|         |

Elected: Lenihan
```

Lenihan was elected, although he received only 27% of the first preference votes.

Here is the two-round runoff result:

```
> tworound.runoff (dublin_west)

|    |Candidate | Total| Percent| ROffTotal| ROffPercent| Elected |
|:---|:---------|-----:|-------:|---------:|-----------:|:-------:|
|1   |Bonnie    |   748|     2.5|         0|         0.0|         |
|2   |Burton    |  3810|    12.7|         0|         0.0|        |
|3   |Ryan      |  2300|     7.7|         0|         0.0|        |
|4   |Higgins   |  6442|    21.5|     12457|        47.3|        |
|5   |Lenihan   |  8086|    27.0|     13900|        52.7|    x   |
|6   |McDonald  |  2404|     8.0|         0|         0.0|        |
|7   |Morrissey |  2370|     7.9|         0|         0.0|        |
|8   |Smyth     |   134|     0.4|         0|         0.0|        |
|9   |Terry     |  3694|    12.3|         0|         0.0|        |
|Sum |          | 29988|   100.0|     26357|       100.0|        |

Elected: Lenihan
```

Lenihan was again elected, but this time after a run-off, as he did not get a majority on the first count. He got an absolute majority on the second count. This indicates a broader base of support than the plurality vote.

We now illustrate the single-winner approval voting method by assuming that voters "approved" any candidate to whom they gave their first, second, or third preference. Under this assumption, voters approved 2.8 candidates on average.

```
> dublin_west2 <- 1*(dublin_west == 1 | dublin_west == 2 | dublin_west == 3)
> approval (dublin_west2)

|    |Candidate | Total| Elected |
|:---|:---------|-----:|:-------:|
|1   |Lenihan   | 15253|    x    |
|2   |Higgins   | 13638|         |
|3   |Burton    | 12863|         |
|4   |Ryan      | 10014|         |
|5   |Terry     |  9810|         |
|6   |Morrissey |  9411|         |
|7   |McDonald  |  6674|         |
|8   |Bonnie    |  4936|         |
|9   |Smyth     |   636|         |
|Sum |          | 83235|         |

Elected: Lenihan
```

Once again, Lenihan wins. The multi-winner approval vote method with three seats gives wins to Lenihan, Higgins, and Burton because they got the most votes.

The Condorcet method did have both a winner and a loser in this case:

```
> condorcet (dublin_west)

|          | Bonnie| Burton| Ryan| Higgins| Lenihan| McDonald| Morrissey| Smyth| Terry| Total| Winner | Loser |
|:---------|------:|------:|----:|-------:|-------:|--------:|---------:|-----:|-----:|-----:|:------:|:-----:|
|Bonnie    |      0|      0|    0|       0|       0|        0|         0|     1|     0|     1|        |       |
|Burton    |      1|      0|    1|       0|       0|        1|         1|     1|     1|     6|        |       |
|Ryan      |      1|      0|    0|       0|       0|        1|         1|     1|     0|     4|        |       |
|Higgins   |      1|      1|    1|       0|       0|        1|         1|     1|     1|     7|        |       |
|Lenihan   |      1|      1|    1|       1|       0|        1|         1|     1|     1|     8|   x    |       |
|McDonald  |      1|      0|    0|       0|       0|        0|         0|     1|     0|     2|        |       |
|Morrissey |      1|      0|    0|       0|       0|        1|         0|     1|     0|     3|        |       |
|Smyth     |      0|      0|    0|       0|       0|        0|         0|     0|     0|     0|        |   x   |
|Terry     |      1|      0|    1|       0|       0|        1|         1|     1|     0|     5|        |       |

Condorcet winner: Lenihan
Condorcet loser: Smyth
```

The STV result is as follows:

```
> stv.dwest <- stv (dublin_west, nseats = 3, eps = 1, digits = 0)

Results of Single transferable vote
===================================
Number of valid votes:   29988
Number of invalid votes:     0
Number of candidates:        9
Number of seats:             3
```

```
|           |       1| 2-trans|     2| 3-trans|      3| 4-trans|        4| 5-trans|       5| 6-trans|         6| 7-trans|    7| 8-trans|      8|
|:----------|-------:|-------:|-----:|-------:|------:|-------:|--------:|-------:|-------:|-------:|---------:|-------:|----:|-------:|------:|
|Quota      |    7498|        |  7491|        |   7486|        |     7465|        |    7303|        |      7233|        | 7043|        |   6143|
|Bonnie     |     748|       8|   756|      20|    776|    -776|         |        |        |        |          |        |     |        |       |
|Burton     |    3810|      55|  3865|       4|   3869|     207|     4076|     295|    4372|     211|      4583|     763| 5345|    1191|   6536|
|Ryan       |    2300|     298|  2598|      23|   2621|      65|     2686|     357|    3042|      77|      3119|     673| 3792|   -3792|       |
|Higgins    |    6442|      68|  6510|      21|   6531|     198|     6728|    1124|    7853|    -550|          |        |     |        |       |
|Lenihan    |    8086|    -588|      |        |       |        |         |        |        |        |          |        |     |        |       |
|McDonald   |    2404|      24|  2428|      19|   2447|      76|     2523|   -2523|        |        |          |        |     |        |       |
|Morrissey  |    2370|      70|  2440|      13|   2453|      98|     2551|     108|    2659|      52|      2711|   -2711|     |        |       |
|Smyth      |     134|       1|   135|    -135|       |        |         |        |        |        |          |        |     |        |       |
|Terry      |    3694|      43|  3737|      21|   3758|      69|     3828|     151|    3979|      71|      4050|     896| 4946|     802|   5748|
|Elected    | Lenihan|        |      |        |       |        |         |        | Higgins|        |          |        |     |        | Burton|
|Eliminated |        |        | Smyth|        | Bonnie|        | McDonald|        |        |        | Morrissey|        | Ryan|        |       |

Elected: Lenihan, Higgins, Burton
```

The three candidates elected were also the ones who got the most first preference votes. All the candidates represented different political parties or were independents, except Ryan and Lenihan, who were both candidates for the Fianna Fáil party, the largest party in Ireland at the time. Lenihan was elected on the first count with a surplus of 588 votes, and 298 of these were transferred to Ryan, the most of any candidate. This reflects the fact that voters tend to give their highest preferences to candidates of the same party, although here, we can see that many of the Lenihan voters did not, in fact, give their second preferences to Ryan.

Although this was an election with almost 30,000 votes and the electoral system appears somewhat complex, the counting takes just two seconds on a Macbook Pro laptop.

Note that the results were slightly different from the results using the Irish STV system, although the same candidates were elected; see Wikipedia (2020b). This is because of several minor differences between the Irish STV system and the `stv` function in the **vote** package. The most important of these is that in the **vote** package, the quota declines as the counts proceed to reflect votes that are not transferred because voters did not express enough preferences. In the Irish STV system, the quota remains the same throughout the counts. We chose to make the quota adaptive because it allows a more complete transfer of the votes of candidates elected. However, if the argument `constant.quota` is set to `TRUE`, the quota is kept constant for all counts.

The STV results can be visualized in several ways. Figure 1 has been produced by the command

```
> plot (stv.dwest)
```

It shows the evolution of the candidate's vote totals over successive counts, as well as of the quota. It can be seen that while candidates mostly stayed in the same order, candidate Ryan overtook two other candidates thanks to transfers, even though she was eventually eliminated. This reflects the fact that she had high preferences among voters who gave their first preferences to Lenihan and Morrissey.

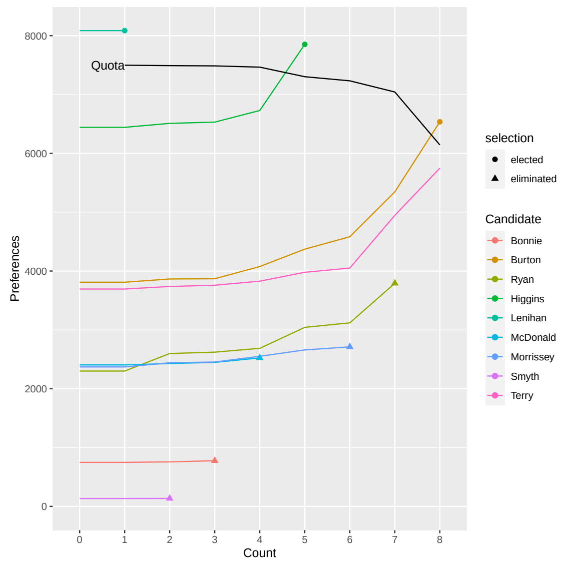


Figure 1: Evolution of candidates' votes over STV counts in the 2002 Irish general election in Dublin West.

Figure 2 shows the number of preference votes that each candidate received. The first preferences reflect the numbers we know from the first count. It can be seen that Ryan and Burton had the most second preferences; in Ryan's case, this is because she was the second Fianna Fáil (FF) candidate behind Lenihan and got the majority of his second preferences. Burton and Morrissey had the most third preferences.

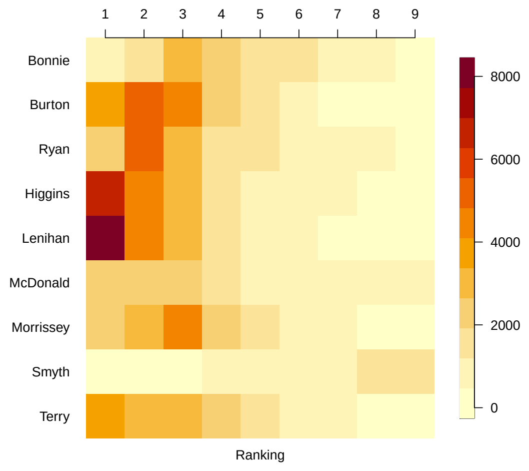


Figure 2: Number of preference votes that each candidate received in the 2002 Irish general election in Dublin West.

Figure 3 (left panel) shows the number of votes for each combination of first and second preference. The biggest number is those who voted first for Lenihan and then for Ryan, again reflecting that they are from the same party and that Lenihan had the most first preferences.

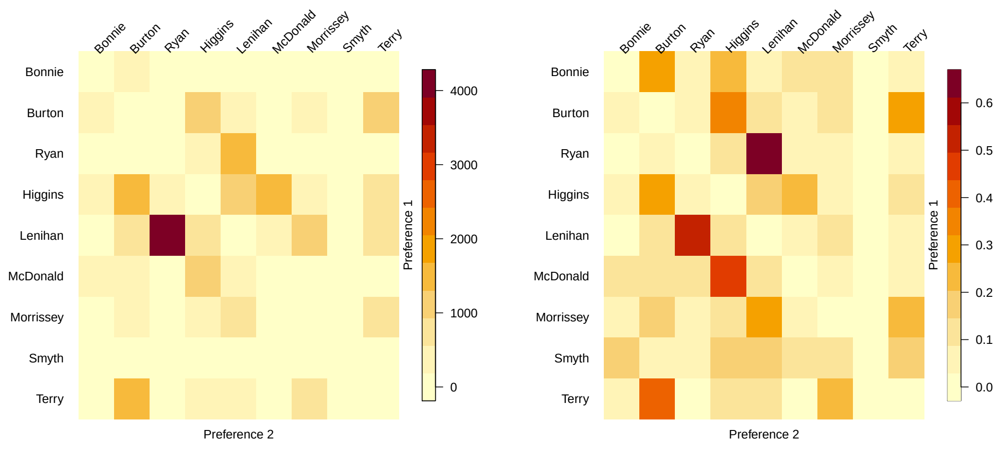


Figure 3: Joint preferences in the 2002 Irish general election in Dublin West. Left panel: Number of votes for each combination of first and second preferences. Right panel: Proportion of the first preference votes for each candidate that gave their second preference vote to each other candidate.

The right panel in Figure 3 shows the same information but in the form of the *proportion* of the first preference voters for each candidate that cast their second preference votes for each other candidate. The largest single cell shows that over 60% of Ryan voters cast their second preferences for Lenihan.

The code for producing Figures 2 and 3 is as follows:

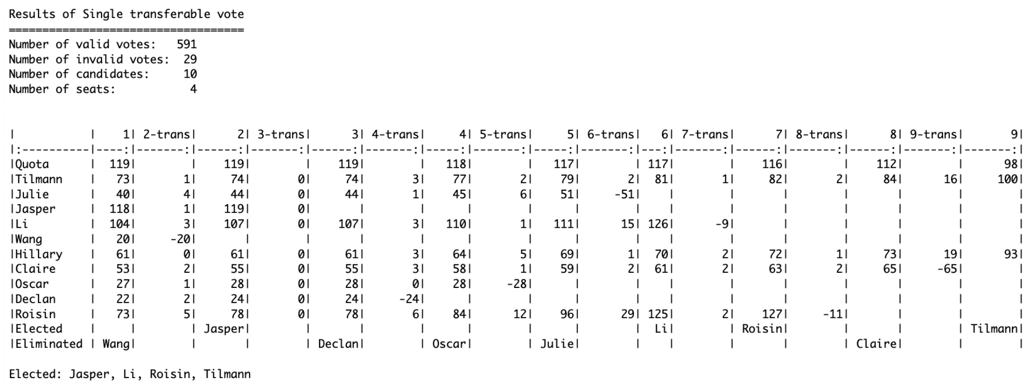
```
> image (stv.dwest, all.pref = TRUE)  # Figure 2
> image (stv.dwest, proportion = FALSE) # Figure 3 left panel
> image (stv.dwest, proportion = TRUE) # Figure 3 right panel
```

Note that the `image` method is available for all functions in the package that use ranked votes, namely `condorcet` and `tworound.runoff`, in addition to `stv`. However, the method cannot be used if equal preferences are present in the ballots.

IMS council election

The `ims_election` dataset contains the votes in a past election for the Council of the Institute of Mathematical Statistics (IMS). There were four seats to be filled with 10 candidates running and 620 voters. The names of the candidates have been anonymized[2]. The election was carried out by STV. The results were:

```
> data (ims_election)
> stv.ims <- stv (ims_election, nseats = 4, eps = 1, digits = 0)
```



```
Results of Single transferable vote
===================================
Number of valid votes:   591
Number of invalid votes:  29
Number of candidates:     10
Number of seats:           4


|           |    1| 2-trans|      2| 3-trans|      3| 4-trans|     4| 5-trans|     5| 6-trans|   6| 7-trans|      7| 8-trans|      8| 9-trans|       9|
|:----------|----:|-------:|------:|-------:|------:|-------:|-----:|-------:|-----:|-------:|---:|-------:|------:|-------:|------:|-------:|-------:|
|Quota      |  119|        |    119|        |    119|        |   118|        |   117|        | 117|        |    116|        |    112|        |      98|
|Tilmann    |   73|       1|     74|       0|     74|       3|    77|       2|    79|       2|  81|       1|     82|       2|     84|      16|     100|
|Julie      |   40|       4|     44|       0|     44|       1|    45|       6|    51|     -51|    |        |       |        |       |        |        |
|Jasper     |  118|       1|    119|       0|       |        |      |        |      |        |    |        |       |        |       |        |        |
|Li         |  104|       3|    107|       0|    107|       3|   110|       1|   111|      15| 126|      -9|       |        |       |        |        |
|Wang       |   20|     -20|       |        |       |        |      |        |      |        |    |        |       |        |       |        |        |
|Hillary    |   61|       0|     61|       0|     61|       3|    64|       5|    69|       1|  70|       2|     72|       1|     73|      19|      93|
|Claire     |   53|       2|     55|       0|     55|       3|    58|       1|    59|       2|  61|       2|     63|       2|     65|     -65|        |
|Oscar      |   27|       1|     28|       0|     28|       0|    28|     -28|      |        |    |        |       |        |       |        |        |
|Declan     |   22|       2|     24|       0|     24|     -24|      |        |      |        |    |        |       |        |       |        |        |
|Roisin     |   73|       5|     78|       0|     78|       6|    84|      12|    96|      29| 125|       2|    127|     -11|       |        |        |
|Elected    |     |        | Jasper|        |       |        |      |        |      |        |  Li|        | Roisin|        |       |        | Tilmann|
|Eliminated | Wang|        |       |        | Declan|        | Oscar|        | Julie|        |    |        |       |        | Claire|        |        |

Elected: Jasper, Li, Roisin, Tilmann
```

The results are shown in Figure 4. Although the electorate was much smaller, the results show some common patterns to those from Dublin West. The quota declined slowly in the early counts and more rapidly in the later ones. The four candidates elected were the ones that got the most first preferences. Figure 4(d) shows that, while there are no political parties in this election, Tilmann and Hillary tended to share voters, as did Jasper and Li. We do not know the identities of the candidates because their names have been anonymized, but these pairs of candidates clearly appeal to the same voters, perhaps because of geographical or intellectual commonalities.

However, neither Li nor Hillary was able to benefit from these shared preferences in this election. While Jasper was elected on the second count, he reached exactly the number of votes needed to reach the quota, namely 119, and thus no surplus was available for a transfer. Tilmann, on the other hand, was elected last, after which the election ended. If there had been one more seat available (i.e., `nseats = 5`), Hillary would have got Tilmann's surplus and then would have been elected.

In this example, 29 votes were identified as invalid. One can explore those votes using

```
> invalid.votes(stv.ims)
```

Most of these votes are all zero preference votes. However, a few of them contain a gap in the ranking. If it is desired that such votes be considered valid up to the last valid ranking, one can add the argument `invalid.partial = TRUE` to the `stv` call. In this case, those votes are corrected. Using `corrected.votes(stv.ims)` will then display the original and corrected versions of the votes. Similarly, `valid.votes(stv.ims)` will display all the valid votes considered in the election.

Trial faculty recruitment vote

This is a trial election that was carried out to test a proposed use of STV in a university statistics department for selecting faculty job candidates to whom to make offers. There were two jobs to be filled, five finalists, and ten voters. It was desired to select the two candidates to whom to make offers and also to produce a ranking of the other candidates. This is fairly typical of such elections. The candidates were named Augustin-Louis Cauchy, Carl Friedrich Gauss, Pierre-Simon Laplace, Florence Nightingale, and Siméon Poisson.

The voters entered their choices into a web-based survey which was then converted into a text file. Here, we create the corresponding dataset manually:

```
> faculty <- data.frame(
+    Cauchy =      c(3, 4, 4, 4, 4, 5, 4, 5, 5, 5),
+    Gauss  =      c(4, 1, 2, 2, 2, 2, 2, 2, 2, 4),
+    Laplace =     c(5, 2, 1, 3, 1, 3, 3, 4, 4, 1),
```

[2]To ensure confidentiality, the names of the candidates were replaced by arbitrarily chosen first names that have no connection to the actual names of the candidates.

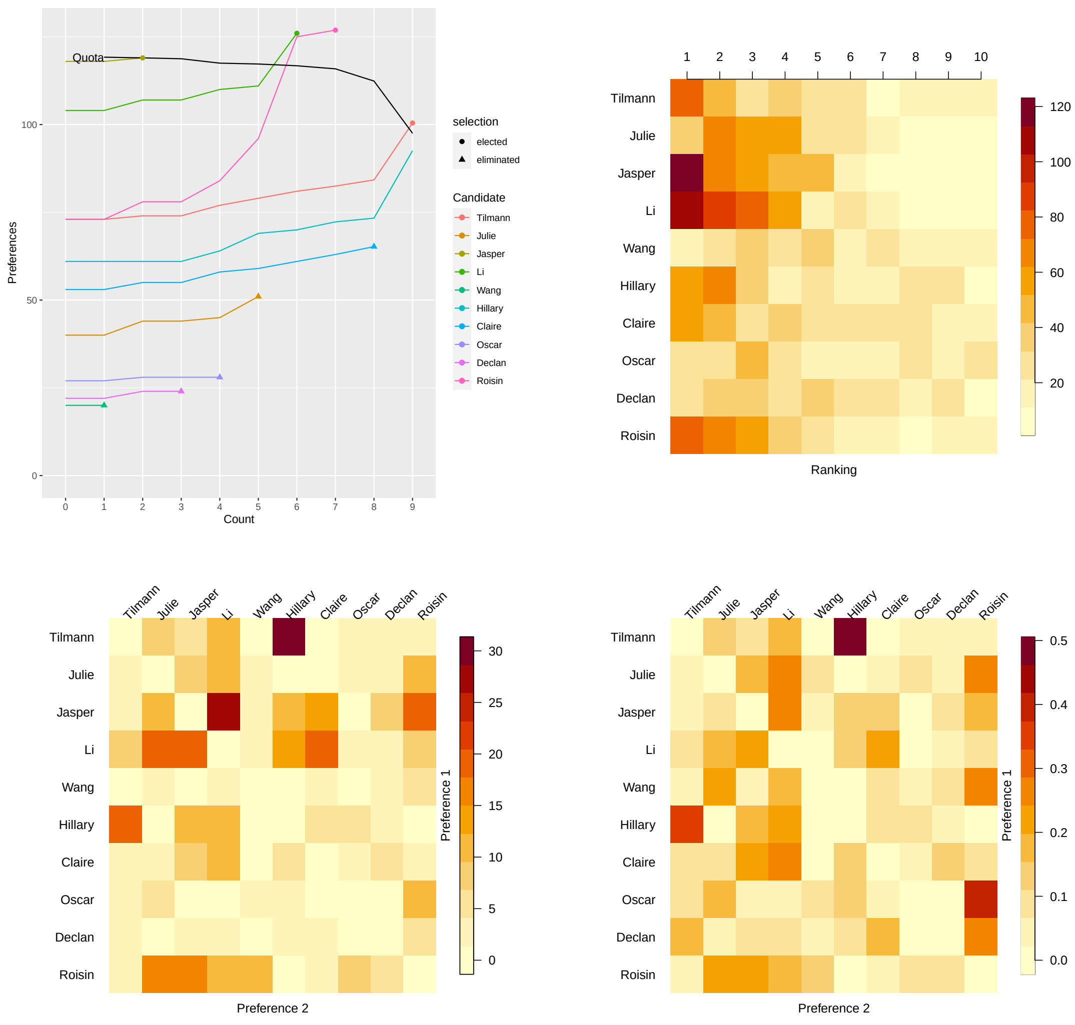


Figure 4: Visualization of results of IMS Council election by STV. (a) Top left: Evolution of votes over counts. (b) Top right: Number of votes for each candidate at each preference level. (c) Bottom left: Number of votes for each first and second preference combination. (d) Bottom right: Number of second preferences as a proportion of the number of first preference voters for each candidate.

```
+    Nightingale = c(1, 3, 5, 1, 3, 1, 5, 1, 1, 2),
+    Poisson =     c(2, 5, 3, 5, 5, 4, 1, 3, 3, 3)
+    )
```

The results of the STV election were as follows:

```
> stv.faculty <- stv (faculty, nseats = 2, digits = 2, complete.ranking = TRUE)

Results of Single transferable vote
===================================
Number of valid votes:   10
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2

|            |           1| 2-trans|      2| 3-trans|       3| 4-trans|     4|
|:-----------|-----------:|-------:|------:|-------:|-------:|-------:|-----:|
|Quota       |        3.33|        |   3.33|        |    3.33|        |  3.33|
|Cauchy      |        0.00|    0.00|   0.00|       0|        |        |      |
|Gauss       |        1.00|    1.33|   2.33|       0|    2.33|    1.33|  3.67|
|Laplace     |        3.00|    0.00|   3.00|       0|    3.00|    0.00|  3.00|
|Nightingale |        5.00|   -1.67|       |        |        |        |      |
|Poisson     |        1.00|    0.33|   1.33|       0|    1.33|   -1.33|      |
|Elected     | Nightingale|        |       |        |        |        | Gauss|
|Eliminated  |            |        | Cauchy|        | Poisson|        |      |

Complete Ranking
================

| Rank|Candidate   | Elected |
|----:|:-----------|:-------:|
|    1|Nightingale |    x    |
|    2|Gauss       |    x    |
|    3|Laplace     |         |
|    4|Poisson     |         |
|    5|Cauchy      |         |

Elected: Nightingale, Gauss
```

Nightingale and Gauss were elected. The complete ranking could be useful for a vote like this, where an ordering beyond the winning candidates may be desired, for example, to make further offers if one of the top two declines the offer. Note that the complete ranking is conditional on the pre-specified number of seats or winners in the election.

The results are illustrated in Figure 5. An interesting feature that can be seen from Figure 5(a) is that Laplace got more first preference votes than Gauss, but Gauss ended up beating him by a small margin for the second offer because almost every voter gave Gauss either their first or second preference. Thus, as other candidates were elected or eliminated, their votes were transferred to Gauss rather than Laplace. The large number of second preferences for Gauss is apparent from Figure 5(b). Figure 5(c) and especially Figure 5(d) show that Gauss got the highest number and proportion of second preference votes from the electors of each of the other candidates.

If this had been done by approval voting, and all the voters had approved their top two choices, the same two candidates would have been selected as by STV (i.e., Nightingale and Gauss).

It is interesting to note that there was no Condorcet winner in this election, even though Nightingale was far ahead of the other candidates by most criteria:

```
> condorcet (faculty)

|            | Cauchy| Gauss| Laplace| Nightingale| Poisson| Total| Loser |
|:-----------|------:|-----:|-------:|-----------:|-------:|-----:|:-----:|
|Cauchy      |      0|     0|       0|           0|       0|     0|   x   |
|Gauss       |      1|     0|       1|           0|       1|     3|       |
|Laplace     |      1|     0|       0|           0|       1|     2|       |
|Nightingale |      1|     1|       0|           0|       1|     3|       |
|Poisson     |      1|     0|       0|           0|       0|     1|       |

There is no condorcet winner (no candidate won over all other candidates).
Condorcet loser: Cauchy
```

This illustrates the fact that even in a relatively clearcut case, there may be no Condorcet winner.

To illustrate the feature of reserved seats in STV, let us assume that it is required that at least one French candidate be selected. Then,

```
> stv (faculty, nseats = 2, group.nseats = 1,
+     group.members = c("Laplace", "Poisson", "Cauchy"), digits = 2)

Results of Single transferable vote
===================================
Number of valid votes:   10
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2
Number of reserved seats:          1
Eligible for reserved seats:       3


|            |           1| 2-trans|      2| 3-trans|       3|
```

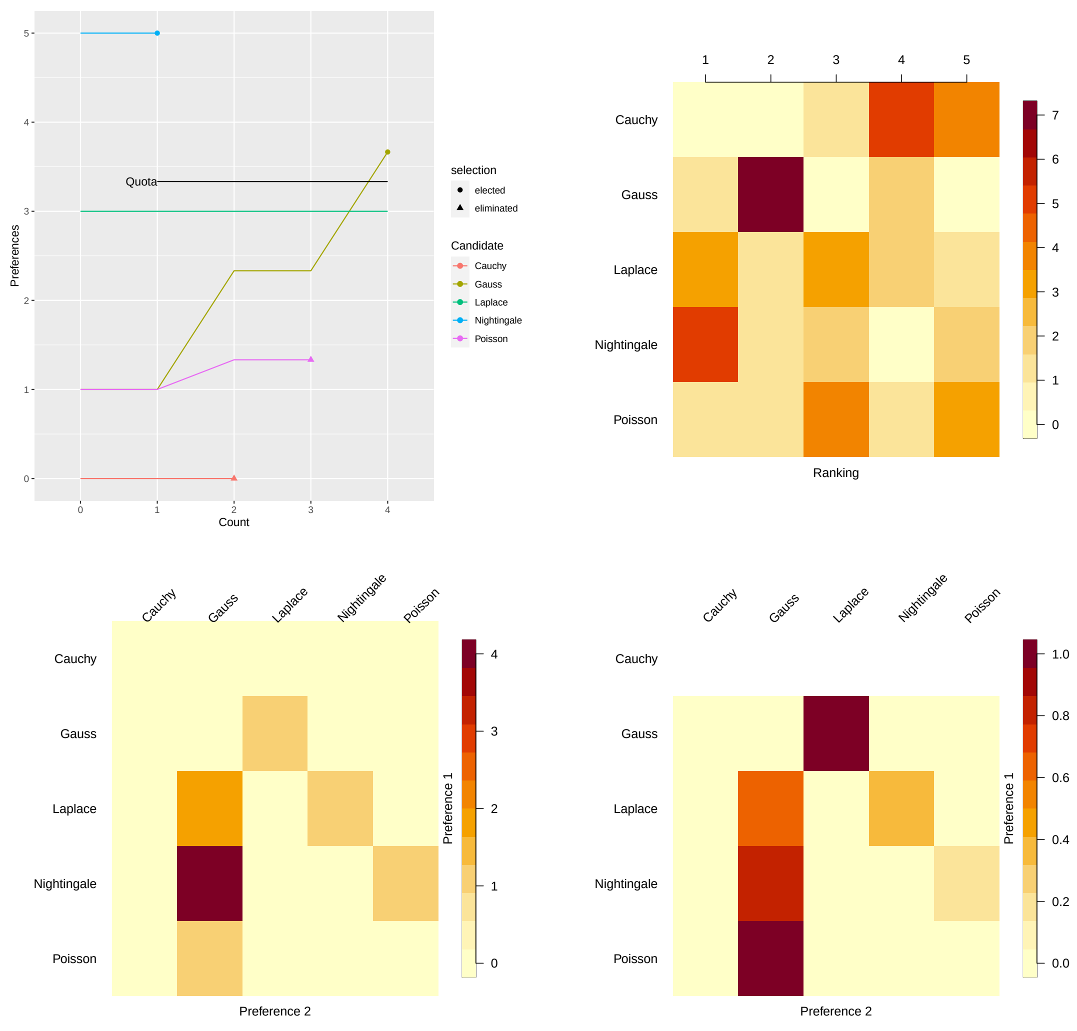


Figure 5: Visualization of results of the trial faculty recruitment vote by STV. (a) Top left: Evolution of votes over counts. (b) Top right: Number of votes for each candidate at each preference level. (c) Bottom left: Number of votes for each first and second preference combination. (d) Bottom right: Number of second preferences as a proportion of the number of first preference voters for each candidate.

```
|:-----------|-----------:|-------:|-----:|-------:|-------:|
|Quota       |        3.33|        |  3.33|        |    3.33|
|Cauchy*     |        0.00|    0.00|  0.00|    0.00|    0.00|
|Gauss       |        1.00|    1.33|  2.33|   -2.33|        |
|Laplace*    |        3.00|    0.00|  3.00|    1.67|    4.67|
|Nightingale |        5.00|   -1.67|      |        |        |
|Poisson*    |        1.00|    0.33|  1.33|    0.67|    2.00|
|Elected     | Nightingale|        |      |        | Laplace|
|Eliminated  |            |        | Gauss|        |        |

Elected: Nightingale, Laplace
```

Here, the modifications to the algorithm described in Section 2.3.5 ensured that none of the French candidates was eliminated on the second count, as the only seat left at that point was the reserved seat. Thus, Gauss, the only non-French candidate left, was eliminated in spite of having more votes than Cauchy or Poisson. Laplace was then elected on the following count. In the output, the candidates eligible for reserved seats are marked with a star.

We now modify this dataset slightly to illustrate the equal ranking STV method. Four of the votes were changed so as to include equal preferences:

```
> faculty2 <- faculty
> faculty2[1,] <- c(2,2,3,1,1)
> faculty2[4,] <- c(3,1,2,1,3)
> faculty2[9,] <- c(4,1,3,1,2)
> faculty2[10,] <- c(2,1,1,1,1)
> faculty2

   Cauchy Gauss Laplace Nightingale Poisson
1       2     2       3           1       1
2       4     1       2           3       5
3       4     2       1           5       3
4       3     1       2           1       3
5       4     2       1           3       5
6       5     2       3           1       4
7       4     2       3           5       1
8       5     2       4           1       3
9       4     1       3           1       2
10      2     1       1           1       1
```

The results of the STV election with equal preferences were as follows:

```
> stv.faculty.equal <- stv (faculty2, equal.ranking = TRUE, digits = 2)
```

```
Results of Single transferable vote with equal preferences
==========================================================
Number of valid votes:   10
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          2

|            |           1| 2-trans|      2| 3-trans|       3| 4-trans|     4|
|:-----------|-----------:|-------:|------:|-------:|-------:|-------:|-----:|
|Quota       |        3.33|        |   3.33|        |    3.33|        |  3.33|
|Cauchy      |        0.00|    0.00|   0.00|       0|        |        |      |
|Gauss       |        2.25|    0.34|   2.59|       0|    2.59|    1.69|  4.28|
|Laplace     |        2.25|    0.01|   2.26|       0|    2.26|    0.13|  2.39|
|Nightingale |        3.75|   -0.42|       |        |        |        |      |
|Poisson     |        1.75|    0.06|   1.81|       0|    1.81|   -1.81|      |
|Elected     | Nightingale|        |       |        |        |        | Gauss|
|Eliminated  |            |        | Cauchy|        | Poisson|        |      |

Elected: Nightingale, Gauss

Warning message:
In correct.ranking(votes, quiet = quiet) :
  Votes 1, 4, 9, 10 were corrected to comply with the required format.
```

The warning message indicates that the ranking was corrected. When `equal.ranking=TRUE`, this correction will be made with any input, as long as the preferences are recorded as positive numbers (not necessarily integers). The corrected votes can be viewed using the `corrected.votes` function, which returns a list. The element new contains the updated votes:

```
> corrected.votes(stv.faculty.equal)$new

   Cauchy Gauss Laplace Nightingale Poisson
1       3     3       5           1       1
4       4     1       3           1       4
9       5     1       4           1       3
10      5     1       1           1       1
```

Such a correction is not made in the default case in which `equal.ranking=FALSE`, when the preferences have to be an ordered sequence of integers starting at one, with no ties and no gaps. However, votes can be corrected in the same way also from outside `stv`, using the function `correct.ranking`. As noted previously, another option is to set `invalid.partial=TRUE` in the `stv` function, which accepts partial-valid ranking, i.e., each vote is considered valid up to its largest valid preference, after which ties and gaps are set to 0. When doing such correction externally via the `correct.ranking` function, set the argument `partial` to `TRUE`.

Finally, we give the results when there is a single winner to illustrate tie-breaking, as it so happens that tie-breaking is needed on two different counts in this case:

```
> stv.faculty.tie <- stv (faculty, nseats = 1)

Results of Single transferable vote
===================================
Number of valid votes:   10
Number of invalid votes:  0
Number of candidates:     5
Number of seats:          1

|            |      1| 2-trans|       2| 3-trans|     3| 4-trans|       4| 5-trans|           5|
|:-----------|------:|-------:|-------:|-------:|-----:|-------:|-------:|-------:|-----------:|
|Quota       |  5.001|        |   5.001|        | 5.001|        |   5.001|        |       5.001|
|Cauchy      |  0.000|       0|        |        |      |        |        |        |            |
|Gauss       |  1.000|       0|   1.000|       1| 2.000|      -2|        |        |            |
|Laplace     |  3.000|       0|   3.000|       0| 3.000|       2|   5.000|      -5|            |
|Nightingale |  5.000|       0|   5.000|       0| 5.000|       0|   5.000|       5|      10.000|
|Poisson     |  1.000|       0|   1.000|      -1|      |        |        |        |            |
|Tie-breaks  |       |        |      fo|        |      |        |       f|        |            |
|Elected     |       |        |        |        |      |        |        |        | Nightingale|
|Eliminated  | Cauchy|        | Poisson|        | Gauss|        | Laplace|        |            |

Elected: Nightingale
```

On the second count, Gauss and Poisson both had one vote, the lowest number, and so were tied for elimination. The Forwards Tie-Breaking method did not break the tie, as they both had the same number of votes also on the first count. The Ordered method did break the tie, however, because Gauss had 7 second preferences, and Poisson had only 1, so Poisson was eliminated. The notation "fo" in the Tie-breaks row indicates the tie-breaking method used, here Forwards followed be Ordered.

On the fourth count, Laplace and Nightingale were tied with 5 votes each, so they were tied for elimination as neither reached the quota of 5.001. The Forwards Tie-Breaking method was then used and involved looking first at their numbers of votes on the first count when Laplace had 3 votes, and Nightingale had 5. As a result, Laplace was eliminated, and then Nightingale was elected. If the Backwards Tie-Breaking method had been used (by setting `ties = "b"`), the comparison would have been done based on the third count instead of the first count. Here too, Laplace had 3 votes, and Nightingale had 5 on the third count, and thus, Laplace would have been eliminated.

Note that the ranking used by the Ordered method can be viewed via the `ordered.tiebreak` function, while passing the valid votes stored in the `data` element of the `stv` object:

```
> ordered.tiebreak(stv.faculty.tie$data)

     Cauchy       Gauss     Laplace Nightingale     Poisson
          1           3           4           5           2
attr(,"sampled")
[1] FALSE FALSE FALSE FALSE FALSE
```

It gives the elimination ranking. When used for electing a candidate, the order is reversed. The attribute "sampled" indicates for each candidate whether sampling was involved in determining its rank, which was not the case in our example. The function `ordered.preferences` can be used to view the matrix of preference counts from which the ordered ranking is derived. It gives the same information as the `image` plot with `all.pref = TRUE`, but in matrix form.

Discussion

We have described and illustrated the **vote** package in R, which implements several electoral systems, namely the plurality, two-round runoff, approval, score, and Single Transferable Vote (STV) systems (Ševčíková et al., 2021). It also identifies the Condorcet winner and loser, if they exist. It implements the Single Transferable Vote system with equal preferences, the first time this has been implemented in software to our knowledge. It also provides several ways of visualizing the STV results.

We are not advocating any electoral system, and indeed it is well known that no one system satisfies all of a set of criteria that one might reasonably want to hold. Thus, which system one uses can depend on the purpose of the election. However, we are particularly interested in multi-winner elections with small electorates, such as committee and council elections in organizations, and the selection of multiple job candidates, award winners, or other choices by small "selectorates." Such elections are common, and there is no universally accepted method for conducting them. We have found the STV system to work well in practice for such elections, and so we have emphasized it here, giving several examples.

For completeness, we note that the most widely used political voting system around the world is a party-list approach, where voters vote for a party rather than for individuals, and some mechanism is then used to fill the party slots allocated (Electoral Reform Society, 2020). Such systems are not relevant for the purposes of our primary interest.

There are several other R packages that implement electoral systems. The **votesys** package implements several electoral methods, including several that are not included in the **vote** package (Wu, 2018). It implements the Instant Runoff Voting (IRV), which is the special case of STV for single-winner elections, but it does not implement the full version of STV for multi-winner elections. The **rcv** package also implements IRV (calling it Ranked Choice Voting) but has been removed from CRAN (Lee and Yancheff, 2019).

The **STV** package implements the STV method (Emerson et al., 2019). The results are generally very similar to those from the `stv` function in the **vote** package. However, there are some minor differences that can lead to different results, particularly in elections with small electorates. Notably, in the **STV** package, all quotas, vote counts, and transfers are rounded to integers, which can lead to different results when the electorate is small. Also, in the **STV** package all tie-breaking is done at random, in contrast with the **vote** package, which uses forwards and backwards tie-breaking. Unlike the **vote** package, none of these other packages implements the STV method with equal ranking or allows for reserved positions for marked groups.

The **HighestMedianRules** implements voting rules electing the candidate with the highest median grade (Fabre, 2020b,a). The **electoral** and **esaps** packages compute various measures of electoral systems; in spite of their names, they do not implement electoral systems or voting rules (Albuja, 2020; Schmidt, 2018).

Acknowledgements: The research of Raftery and Ševčíková was supported by NIH grant R01 HD070936. The authors are grateful to Salvatore Barbaro and Brendan Murphy for helpful discussions and to an anonymous reviewer for helpful comments that improved the article.

Bibliography

Ace Project. The systems and their consequences, 2020. URL `https://aceproject.org/ace-en/topics/es/esd/default`. [p674]

J. Albuja. *electoral: Allocating Seats Methods and Party System Scores*, 2020. URL `https://CRAN.R-project.org/package=electoral`. R package version 0.1.2. [p694]

K. Arrow. *Social Choice and Individual Values*. Yale University Press, New Haven, Conn., 2nd edition, 1963. [p674]

S. J. Brams and P. C. Fishburn. Approval voting. *American Political Science Review*, 72:831–847, 1978. [p676]

S. J. Brams and P. C. Fishburn. *Approval Voting*. Birkhauser, Boston, Mass., 1983. [p677]

S. J. Brams and P. C. Fishburn. *Approval Voting*. Springer Science & Business Media, 2nd edition, 2007. [p676]

M. de Condorcet. *Essai sur l'Application de l'Analyse à la Probabilité des Decisions Rendues à la Pluralité des Voix*. Imprimerie Royale, Paris, France, 1785. [p677]

Electoral Reform Society. Party list proportional representation, 2020. URL `https://www.electoral-reform.org.uk/voting-systems/types-of-voting-system/party-list-pr`. [p694]

J. Emerson, S. Chandra, and L. Orr. *STV: Single Transferable Vote Counting*, 2019. URL `https://CRAN.R-project.org/package=STV`. R package version 1.0.1. [p694]

A. Fabre. Tie-breaking the highest median: alternatives to the majority judgment. *Social Choice and Welfare*, 2020a. doi: https://doi.org/10.1007/s00355-020-01269-9. [p694]

A. Fabre. *HighestMedianRules: Implementation of Voting Rules Electing the Candidate with Highest Median Grade*, 2020b. URL `https://CRAN.R-project.org/package=HighestMedianRules`. R package version 1.0. [p694]

Fair Vote. Multi-winner ranked choice voting, 2020. URL `https://www.fairvote.org/multi_winner_rcv_example`. [p678]

A. Gibbard. Manipulation of voting schemes: A general result. *Econometrika*, 41:587–601, 1973. [p674]

I. D. Hill. Some aspects of elections – To fill one seat or many (with discussion). *Journal of the Royal Statistics Society, Series A (Statistics in Society)*, 151:243–275, 1988. [p677, 678]

I. D. Hill, B. A. Wichmann, and D. R. Woodall. Single transferable vote by Meek's method. *Computer Journal*, 30:277–281, 1987. [p673, 679]

E. Kitchener. A new way to break STV ties in a special case. *Voting Matters*, 20:9–11, 2005. URL `http://www.votingmatters.org.uk/ISSUE20/I20P3.PDF`. [p680]

J. Lee and M. Yancheff. *rcv: Ranked Choice Voting*, 2019. URL `https://cran.r-project.org/src/contrib/Archive/rcv/`. R package version 0.2.1. [p694]

London Elects. Mayor of London & London Assembly elections: Counting the votes, 2020. URL `https://www.londonelects.org.uk/im-voter/counting-votes`. [p676]

J. Lundell. Random tie-breaking in STV. *Voting Matters*, 22:1–6, 2006. URL `http://www.votingmatters.org.uk/ISSUE22/I22P1.pdf`. [p680]

B. L. Meek. Une nouvelle approche du scrutin transférable. *Mathématiques et Sciences Humaines*, 25: 13–23, 1969. [p679]

B. L. Meek. Une nouvelle approche du scrutin transférable (fin). *Mathématiques et Sciences Humaines*, 29:33–39, 1970. [p679, 682]

R. A. Newland, F. S. Britton, C. Rosenstiel, and J. Woodward-Nutt. *How to Conduct an Election by the Single Transferable Vote*. Electoral Reform Society, London, U.K., 1997. URL `https://www.electoral-reform.org.uk/latest-news-and-research/publications/how-to-conduct-an-election-by-the-single-transferable-vote-3rd-edition`. [p678, 680]

J. C. O'Neill. Tie-breaking with the single transferable vote. *Voting Matters*, 18:14–17, 2004. URL `http://www.votingmatters.org.uk/ISSUE18/I18P6.PDF`. [p680]

M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975. [p674]

N. Schmidt. *esaps: Indicators of Electoral Systems and Party Systems*, 2018. URL `https://CRAN.R-project.org/package=esaps`. R package version 0.1.0. [p694]

H. Ševčíková, A. E. Raftery, and B. W. Silverman. *vote: Election Vote Counting*, 2021. URL `https://CRAN.R-project.org/package=vote`. R package version 2.3-0. [p673, 693]

B. W. Silverman. Single transferable voting system: Counting IMS elections by single transferable vote, 2002. URL `https://imstat.org/elections/single-transferable-voting-system`. [p673, 679, 682]

B. W. Silverman. Counting IMS elections: new procedure. *IMS Bulletin*, 32:3, 2003. [p673, 679]

E. Syddique. Discussion of 'Some aspects of elections – To fill one seat or many'. *Journal of the Royal Statistical Society, Series A (Statistics in Society)*, 151:265, 1988. [p674]

N. Tideman. The single transferable vote. *Journal of Economic Perspectives*, 9:27–38, 1995. [p678]

Wikipedia. Condorcet method, 2020a. URL `https://en.wikipedia.org/wiki/Condorcet_method`. [p678]

Wikipedia. Dublin West Dáil constituency, 2020b. URL `https://en.wikipedia.org/wiki/Dublin_West#2002_general_election`. [p686]

Wikipedia. Single transferable vote, 2020c. URL `https://en.wikipedia.org/wiki/Single_transferable_vote`. [p674, 682]

J. Wu. *votesys: Voting Systems, Instant-Runoff Voting, Borda Method, Various Condorcet Methods*, 2018. URL `https://CRAN.R-project.org/package=votesys`. R package version 0.1.1. [p694]

Adrian E. Raftery
Departments of Statistics and Sociology
University of Washington
Box 354322
Seattle, WA 98195-4322, USA
ORCID: 0000-0002-6589-301X
raftery@uw.edu

Hana Ševčíková
Center for Statistics and the Social Sciences
University of Washington
Box 354320
Seattle, WA 98195-4329, USA
ORCID: 0000-0002-7896-1704
hanas@uw.edu

Bernard W. Silverman
Rights Lab
University of Nottingham
Highfield House
University Park
Nottingham, NG7 2RD, U.K.
ORCID: 0000-0002-4059-2376
bernard.silverman@stats.ox.ac.uk

RPESE: Risk and Performance Estimators Standard Errors with Serially Dependent Data

by Anthony-Alexander Christidis and R. Douglas Martin

Abstract The R package **RPESE** (Risk and Performance Estimators Standard Errors) implements a new method for computing accurate standard errors of risk and performance estimators when returns are serially dependent. The new method makes use of the representation of a risk or performance estimator as a summation of a time series of influence-function (IF) transformed returns, and computes estimator standard errors using a sophisticated method of estimating the spectral density at frequency zero of the time series of IF-transformed returns. Two additional packages used by **RPESE** are introduced, namely **RPEIF** which computes and provides graphical displays of the IF of risk and performance estimators, and **RPEGLMEN** which implements a regularized Gamma generalized linear model polynomial fit to the periodogram of the time series of the IF-transformed returns. A Monte Carlo study shows that the new method provides more accurate estimates of standard errors for risk and performance estimators compared to well-known alternative methods in the presence of serial correlation.

Introduction

In the risk and portfolio management industry, risk and performance estimators are standard tools used in data-driven decision-making processes. The current industry practice in reporting risk and performance estimates for individual assets and portfolios typically do not include their standard error (SE) estimates. For this reason, consumers of such reports have no way of knowing the statistical accuracy of the estimates. As a leading example, one seldom sees SEs reported for Sharpe ratios, and consequently cannot tell whether or not two Sharpe ratios for two different portfolios or assets are significantly different. This motivated the development of the Risk and Performance Estimator Standard Errors package, **RPESE**, for computing risk and performance estimator standard errors that are accurate when returns that are serially correlated and can have fat-tailed and skewed non-normality of returns distributions. **RPESE** uses a new method for computing risk and performance estimators standard errors developed by Chen and Martin (2021).

In the quantitative finance literature, numerous statistical methods have been proposed to evaluate the variability of risk and performance estimators under the assumption of independent and identically distributed (i.i.d.) returns. These methods are however inadequate when returns are serially correlated. An example of this was provided by Lo (2002), who showed that the Sharpe ratio estimator standard error based on the assumption of i.i.d. returns is overly optimistic when returns are actually serially correlated. In terms of statistical methods to compute standard errors of estimators when working with time series data exhibiting serial correlation, there are two well-known methods in the literature: the nonparametric heteroskedasticity and autocorrelation consistent (HAC) covariance method (Newey and West, 1987; Andrews, 1991; Zeileis, 2004) and the nonparametric block bootstrap method (Kunsch, 1989; Politis and Romano, 1994). HAC covariance estimators, which rely on a kernel-based approach on the lag-k covariance estimates of the time series, possess desirable statistical properties such as consistency under heteroscedasticity and autocorrelation. Block bootstrap methods have been shown to provide accurate standard errors if they are well calibrated (Ledoit and Wolf, 2008). However, they rely on randomness which is an unappealing feature for industry practitioners as different results can be reported on the same data set, albeit small if a large enough number of replicates are used.

Unlike the nonparametric HAC and block bootstrap methods, the method implemented in the **RPESE** package is a fully parametric and deterministic method for computing risk and performance estimators standard errors for time series with serial correlation, which is an appealing feature for the purpose of constructing confidence intervals and tests. While IFs are commonly used tools in robust statistics, they are seldomly used in quantitative finance research and applications. The new method involves the representation of the risk and performance estimators using influence functions (IFs) and fitting a penalized Gamma generalized linear model (GLM) to the periodogram of the IF-transformed returns time series. The estimate of the variance of the estimator can be obtained from the spectral density estimate of the IF-transformed returns time series at zero frequency, a technique introduced by Heidelberger and Welch (1981).

The remainder of this article is organized as follows. The "New Methodology: Standard Errors via Influence Functions" section provides the theoretical and computational details of the new method to compute standard errors of risk and performance estimators. The "Influence Functions for Risk and

Performance Estimators" section presents five risk estimators and seven performance estimators with their influence functions, and introduces the **RPEIF** package with some sample code for the purpose of computing and providing graphical displays of such influence functions. The "Periodogram Based Generalized Linear Model Method" section describes the computational details of the Gamma GLM fit to the periodogram of the IF-transformed returns time series and briefly discusses the implementation in the **RPEGLMEN** package. The "Application: Hedge Funds Data Standard Errors with **RPESE**" section provides example code to compute standard errors for hedge funds returns data. The "Monte Carlo Study with IF-Based Standard Errors" section presents a benchmark comparison of the new method with two HAC based approaches for serially correlated data. The "Summary" section discusses future developments for the new method and the **RPESE** package.

New Methodology: Standard Errors via Influence Functions

IFs were first introduced by Hampel (1974), and were developed further by Hampel et al. (1986). In this section, we provide the definition and basic properties of influence functions, with a view toward their use for understanding the influence of outliers on risk and performance estimators, and for computing standard errors of such estimators for both uncorrelated and serially correlated returns.

Risk and Performance Estimator Functional Representations

The large-sample value (as sample size n tends to infinity) of a risk or performance estimator may be represented as a functional $T = T(F)$ of the marginal distribution function F of a time series $r_1, r_2, \ldots, r_n$ of returns.[1] For example the functional for the mean (expected value) is

$$\mu(F) = \int r dF(r) \tag{1}$$

and the functional for the standard deviation (returns volatility) is

$$\sigma(F) = \left[\int (r - \mu(F))^2 dF(r) \right]^{\frac{1}{2}}. \tag{2}$$

Given a functional representation $T(F)$ of an estimator, a finite-sample *non-parametric* estimator T_n is easily obtained by replacing the unknown distribution F by the empirical distribution F_n that has a jump of height $1/n$ at each of the observed returns values $r_1, r_2, \ldots, r_n$:

$$T_n = T(F_n) = T(r_1, r_2, \ldots, r_n). \tag{3}$$

For example, the finite-sample non-parametric estimators of the mean and standard deviation are the sample mean and sample standard deviation, respectively:

$$\hat{\mu}_n = \frac{1}{n} \sum_{t=1}^{n} r_t \qquad \hat{\sigma}_n = \left[\frac{1}{n} \sum_{t=1}^{n} (r_t - \hat{\mu}_n) \right]^{\frac{1}{2}}. \tag{4}$$

We note that one can also derive parametric estimators from parametric functional representation obtained by replacing F by F_θ, where θ is the parameter vector for a parametric distribution function. In this case one obtains the finite-sample estimator by replacing the unknown parameter by its estimator, typically the maximum-likelihood estimator (MLE). See for example, Martin and Zhang (2019) for a treatment of parametric and non-parametric expected shortfall (ES) estimators for normal and t-distributions. However, the current version of the **RPEIF** package only deals with non-parametric risk and performance estimators.

Estimator Influence Function Definition

IFs are based on the use of the following mixture distribution perturbation of a fixed target distribution $F(x)$:

$$F_\gamma(x) = (1 - \gamma)F(x) + \gamma\delta_r(x),\ 0 \le \gamma < 1/2 \tag{5}$$

[1]The term functional refers to a function whose domain is an infinite dimensional space, e.g., the space of distribution functions.

where $\delta_r(x)$ is a point mass discrete distribution function with a jump of height one located at value r . The IF of an estimator with functional form $T(F)$ is defined as:

$$IF(r; T, F) = \lim_{\gamma \to 0} \frac{T(F_\gamma) - T(F)}{\gamma} = \frac{d}{d\gamma} T(F_\gamma)|_{\gamma=0} \tag{6}$$

The IF is a special directional derivative (i.e., a Gateaux derivative) of the functional $T(F)$ in the direction of a point mass distributions δ_r, evaluated at F. It is straightforward, and more or less tedious, to derive formulas for the IFs of risk and performance estimators. For example, the IF of the sample mean is:

$$IF(r; \mu; F) = r - \mu \tag{7}$$

where $\mu = \mu(F)$ depends on the underlying returns marginal distribution F. The above influence function has the property that its expected value is zero, which is a reflection of the general property than an influence function has zero expected value (Hampel, 1974):

$$E\,[IF(r; T, F)] = 0. \tag{8}$$

A Key Influence Function Property

A key IF property is that for well behaved estimator functionals, the finite-sample estimator $T_n = T(F_n) = T(r_1, r_2, \ldots, r_n)$ can be expressed in terms of the sample mean of IF transformed returns as

$$T_n - T(F) = \frac{1}{n} \sum_{t=1}^{n} IF(r_t; T, F) + remainder \tag{9}$$

where the remainder goes to zero in a probabilistic sense as $n \to \infty$. Thus the finite sample variance of T_n is approximately given by

$$Var\,(T_n) = Var\left[\frac{1}{n} \sum_{t=1}^{n} IF(r_t; T, F)\right] \tag{10}$$

and in the special case where the returns r_t are i.i.d., the IF-transformed returns are i.i.d., and the variance of T_n reduces to

$$Var\,(T_n) = \frac{1}{n} E[IF^2(r_1; T, F)]. \tag{11}$$

and the expectation on the right-hand side can be evaluated empirically as the sample mean of the squared influence functions.

However, when the $r_t, t = 1, 2, \ldots, n$ are serially dependent, such as in the case of serially correlated AR(1) returns or serially uncorrelated but dependent GARCH(1,1) returns, the IF-transformed returns time series $IF(r_t; T, F)$ will generally have serial correlation that needs to be accounted for in calculating the variance on the the right-hand-side of (10). In particular,

$$\frac{1}{n} Var\left[\sum_{i=1}^{n} IF(r_t; T, F)\right] = C_{IF}(0) + 2 \sum_{k=1}^{n-1} \left(1 - \frac{k}{n}\right) C_{IF}(k), \tag{12}$$

where $C_{IF}(k)$ is the symmetric lag-k covariance between $IF(r_t)$ and $IF(r_{t+k})$. Spectral analysis theory, extensively used in science and engineering, shows that the variance of the sum of the values of a serially correlated stationary time series is given by the spectral density of the time series at zero frequency. Thus the problem of estimating the variance (10), with possibly serially correlated returns, reduces to the problem of estimating the spectral density at zero frequency of the the time series $IF(r_t; T, F)/n$. Chen and Martin (2021) show how to do this by a polynomial Gamma GLM fitting method, with elastic net (EN) regularization, that works well when the returns are serially correlated, as well as when they are uncorrelated. Their methodology is implemented in the **RPESE** package, which in turn makes fundamental use of the **RPEIF** package to compute and provide graphical displays for the influence functions of popular risk and performance estimators, and the **RPEGLMEN** to fit a polynomial Gamma GLM with EN penalty to the IF-transformed returns time series periodogram.

Influence Functions for Risk and Performance Estimators

The current version (1.2.2) of the **RPESE** package supports six risk and nine performance estimators. The risk estimators are the sample standard deviation (SD), the semi-standard deviation (SemiSD), the lower partial moment of order one and two (LPM1 and LPM2), the expected shortfall (ES) and the

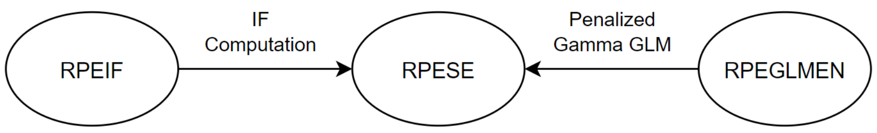


Figure 1: Package relations between RPEIF, RPESE and RPEGLMEN.

value-at-risk (VaR), the latter two risk estimators with tail probability α. The performance estimators are the mean (Mean), a robust M-estimator of the mean (robMean) of ψ-type[2], the Sharpe ratio (SR), the downside Sharpe ratio (DSR), the Sortino ratio (SoR) with constant threshold c, the expected shortfall ratio (ESratio) and the value-at-risk ratio (VaRratio) with tail probability α, the Rachev ratio (RachevRatio) with lower and upper tail probability α and β, and the Omega ratio (OmegaRatio) with constant threshold c.

The formulas for these risk and performance estimators and their functional representations, and the derivations of their influence function formulas, are given in Zhang et al. (2021). In Table 1, the names of the functions in the RPEIF and RPESE packages for the risk and performance estimators are provided.

Risk	**RPEIF**	**RPESE**
SD	`IF.SD`	`SD.SE`
SemiSD	`IF.SemiSD`	`SemiSD.SE`
LPM1 or LPM2	`IF.LPM`	`LPM.SE`
ES	`IF.ES`	`ES.SE`
VaR	`IF.VaR`	`VaR.SE`
Performance	**RPEIF**	**RPESE**
Mean	`IF.Mean`	`Mean.SE`
robMean	`IF.robMean`	`robMean.SE`
SR	`IF.SR`	`SR.SE`
DSR	`IF.DSR`	`DSR.SE`
SoR	`IF.SoR`	`SoR.SE`
ESratio	`IF.ESratio`	`Esratio.SE`
VaRratio	`IF.VaRratio`	`VaRratio.SE`
RachevRatio	`IF.RachevRatio`	`RachevRatio.SE`
OmegaRatio	`IF.OmegaRatio`	`OmegaRatio.SE`

Table 1: Functions in RPEIF and RPESE for the risk and performance estimators.

The RPEIF Package

The functions in the RPEIF package listed in Table 1 are used for two distinct purposes. The first is to evaluate an estimator IF at a set of data values, and plot them to display a graph of the influence function. This allows the user to explore the different shapes of the IFs of different estimators. The second and primary purpose of these IF functions is to compute IF-transformed time series of returns, as a first step in the overall method of computing standard errors for risk and performance estimators.

To briefly demonstrate the usage of the RPEIF package, the edhec data set available in the PerformanceAnalytics package will be used. This data set contains hedge fund returns from January, 1997 to November, 2019. The data can be loaded as an `xts` time series object with the following R code.

```
install.packages("PerformanceAnalystics")
data(edhec, package = "PerformanceAnalytics")
class(edhec)

## [1] "xts" "zoo"
```

The hedge fund names in edhec are too long to display well in plots. The following code replaces those long names with shorter names.

[2]Commonly referred to as a robust M-estimator of "location" in the literature.

```
colnames(edhec) <- c("CA", "CTAG", "DIS", "EM", "EMN", "ED", "FIA", "GM", "LS", "MA",
                     "RV", "SS", "FoF")
```

The following functions are available in RPEIF:

```
library(RPEIF)
ls("package:RPEIF")
```

```
 [1] "IF"             "IF.DSR"         "IF.ES"          "IF.ESratio"
 [5] "IF.LPM"         "IF.Mean"        "IF.OmegaRatio"  "IF.RachevRatio"
 [9] "IF.robMean"     "IF.SD"          "IF.SemiSD"      "IF.SoR"
[13] "IF.SR"          "IF.VaR"         "IF.VaRratio"    "nuisParsFn"
```

In order to compute the values and plot the shapes of influence functions using the IF functions in Table 1, nuisance parameters need to be specified, and there are two basic methods of doing so. Using the nuisParsFn function, nuisance parameters can be generated by specifying "typical" values based on some assumed returns distribution. For the risk measure estimators and performance measure estimators, the nuisParsFn function assumes by default that the returns follow the normal distribution, with monthly mean return of $\mu = 1\%$, risk-free rate $rf = 0\%$, monthly volatility of $\sigma = 5\%$ (the corresponding annual mean and volatility are 12% and 17.3%, respectively), and in addition assumes by default that $c = 0$ for LPM and SoR, $\alpha = 0.10$ for VaR and ES, and in addition $\beta = 0.10$ for the Rachev ratio. Thus:

```
args(nuisParsFn)
```

```
## function (mu = 0.01, sd = 0.05, c = 0, alpha = 0.1, beta = 0.1)
```

To generate nuisance parameters by using a mean return of 2% instead of 1% and a volatility of 15% instead of 5% (the defaults) for the purpose of displaying the IF plots for the SD and the SR, the nuisPars argument can be used as in the following code, with the plots shown in Figure 2.

```
par(mfrow = c(2, 1))
outSD <- IF.SD(evalShape = T, IFplot = T, nuisPars = nuisParsFn(mu = 0.02, sd = 0.15))
outSR <- IF.SR(evalShape = T, IFplot = T, nuisPars = nuisParsFn(mu = 0.02, sd = 0.15))
```

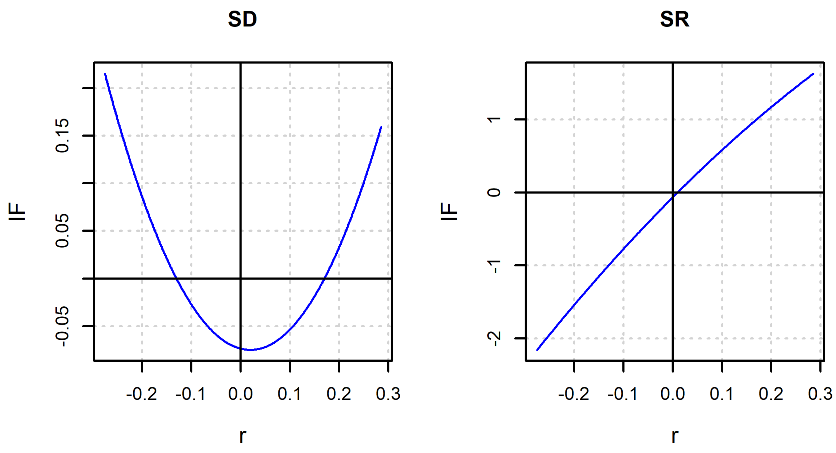


Figure 2: IF shapes of SD and SR with user-specified nuisance parameters.

The second way to specify nuisance parameter values is to estimate them from a returns time series of interest, the latter of which is specified by using the argument returns of the IF functions. For example, the Convertible Arbitrage (CA) hedge fund time series can be used for this purpose by using the following code, the results of which are shown in Figure 3.

```
par(mfrow = c(2, 1))
outSD <- IF.SD(returns = edhec$CA, evalShape = T, IFplot = T)
outSR <- IF.SR(returns = edhec$CA, evalShape = T, IFplot = T)
```

To plot the IF-transformed returns of the SD and SR estimators, use the following code to generate the output shown in Figure 4:

```
SDiftr <- IF.SD(returns = edhec$CA)
SRiftr <- IF.SR(returns = edhec$CA)
```

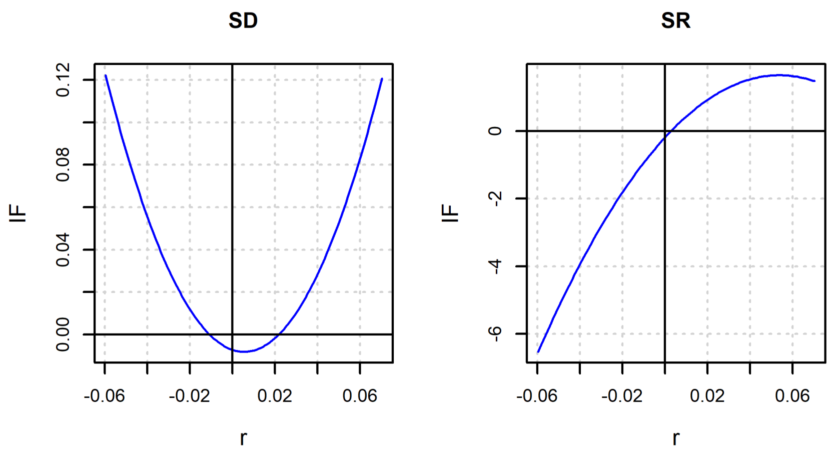


Figure 3: IF shapes of SD and SR with estimated nuisance parameters.

```
par(mfrow = c(3, 1))
plot(edhec$CA, lwd = 0.8, ylab = "Returns", main = "CA Hedge Fund Returns")
plot(SDiftr, lwd = 0.8, main = "IF.SD Transformed Returns")
plot(SRiftr, lwd = 0.8, main = "IF.SR Transformed Returns")
```

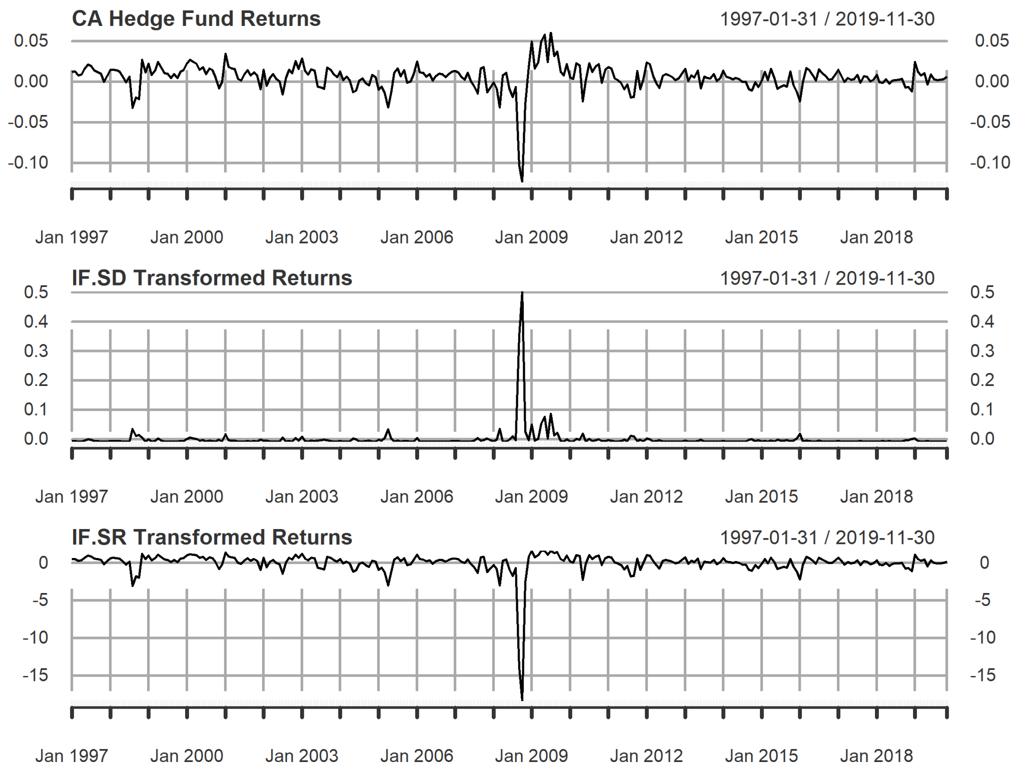


Figure 4: SD and SR IF-transformed returns for CA hedge fund.

Returns data are prone to outliers which adversely influence parameter estimates and inflate the standard errors of risk and performance estimators. A very reliable outlier cleaning method that shrinks outliers can be obtained based on a robust location M-estimator and an associated robust scale estimator $\hat{s}$. A location M-estimator is computed as a solution of the equation

$$\sum_{t=1}^{n} \psi_{mOpt}\left(\frac{r_t - \hat{\mu}_M}{\hat{s}}\right) = 0 \tag{13}$$

where $\psi_{mOpt} = \psi_{mOpt}(x)$ is an optimal bias robust "psi" function, and $\hat{s}$ is a robust scale estimate of the residuals $\epsilon_t = r_t - \hat{\mu}_M$. For an introduction to location M-estimators and their computation, see

Sections 2.3 and 2.7 of Maronna et al. (2019). The optimal bias robust function ψ_{mOpt} is the default used by the function `locScaleM` in the **RobStatTM** package. The `robMean` function in **RPESE** computes the estimates $\hat{\mu}_M$ and $\hat{s}$ using `locScaleM`.

Based on a location estimate $\hat{\mu}_M$ and associated scale estimate $\hat{s}$, it is natural to define returns r_t that fall outside the interval $[\hat{\mu}_M - 3 \cdot \hat{s}, \hat{\mu}_M + 3 \cdot \hat{s}]$ as outliers for a 95% efficiency. Such outliers are then "cleaned" by shrinking them to the nearest boundary of that interval. For the Fixed Income Arbitrage (FIA) hedge fund returns, using the optimal bias robust ψ_{mOpt} function, it turns out that the above interval is $[-0.01171, 0.02231]$. Correspondingly, the FIA hedge fund returns with values less than -0.01171, or greater than 0.02231, are detected as outliers and shrunk accordingly.

The following code results in Figure 5, which shows the sample mean IF-transformed FIA returns (which are equal to FIA returns minus the very small mean of the FIA returns) in the top plot, and the outlier cleaned IF tranformaed FIA returns in the bottom plot.

```
iftrFIA <- IF.Mean(returns = edhec$FIA)
iftrFIAclean <- IF.Mean(returns = edhec$FIA, cleanOutliers = T)
par(mfrow = c(2, 1))
plot(iftrFIA, main = "FIA IF-transformed Returns", lwd = 0.8)
plot(iftrFIAclean, main = "Outlier Cleaned FIA IF-transformed Returns", lwd = 0.8)
```

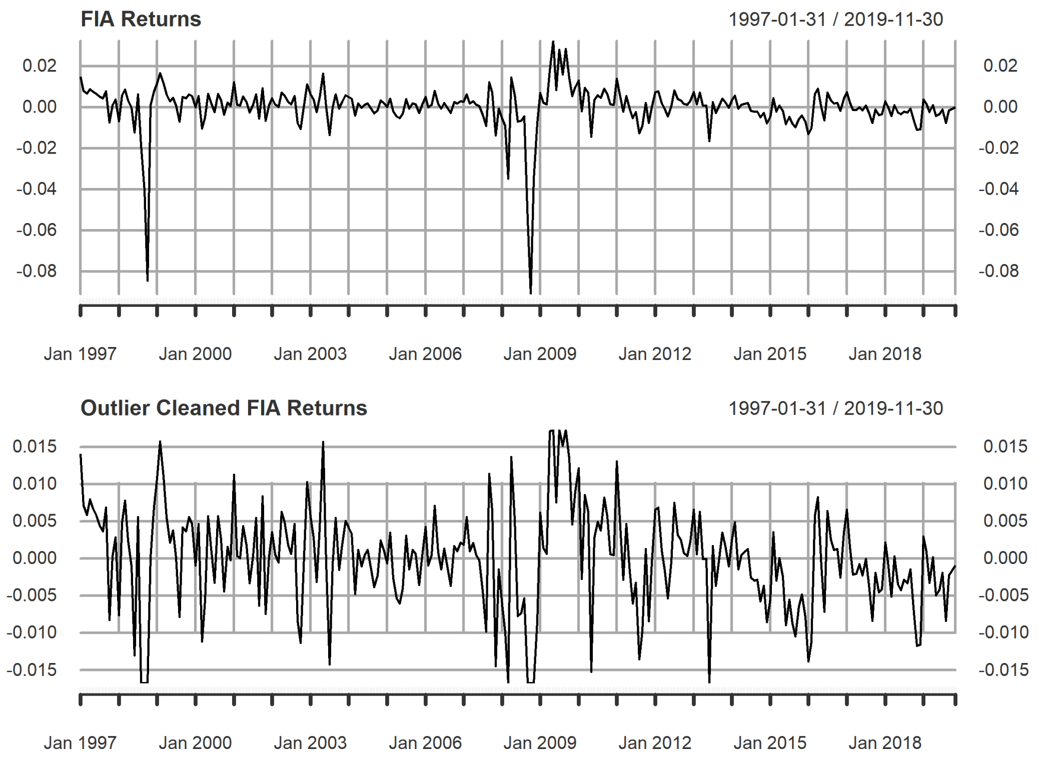


Figure 5: IF and outlier cleaned IF-transformed FIA returns.

Spectral density function estimation is a frequently used method in the field of signal processing, and in other engineering and science applications. Prewhitening is a technique often used to improve the performance of spectral density estimators. Since the core of the method described in Chen and Martin (2021) is estimation of a spectral density at frequency zero of an IF-transformed returns time series IF_t, it is natural to be able to use prewhitening of that time series. The prewhitened variant of the basic IF-based SE method in the **RPESE** package implement such prewhitening. A prewhitened version IF_t^{pw} of the IF_t time series by using the prewhitening transformation

$$IF_t^{pw} = IF_t - \hat{\rho} IF_{t-1} \tag{14}$$

where $\hat{\rho}$ is a lag-one serial correlation coefficient estimat of the IF_t. In general the IF_t^{pw} series is not a serially uncorrelated (white noise) series, but it has considerably less serial correlation than IF_t, and a periodogram estimator based on IF_t^{pw} will suffer from relatively little bias compared with one based on IF_t. Since outliers can have adverse influence not only on risk and performance estimators, but also on the estimator $\hat{\rho}$ used for prewhitening, it is always a good idea to clean outliers before prewhitening. This can done for example using the following code, which results in the plot shown in Figure 6.

```
iftrFIAcl <- IF.Mean(returns = edhec$FIA, cleanOutliers = T)
PWiftrFIAcl <- IF.Mean(returns = edhec$FIA, cleanOutliers = T, prewhiten = T)
par(mfrow = c(2, 1))
plot(iftrFIAcl, main = "FIA Outlier Cleaned Returns", lwd = 0.8)
plot(PWiftrFIAcl, main = "Prewhitened FIA Outlier Cleaned Returns", lwd = 0.8)
```

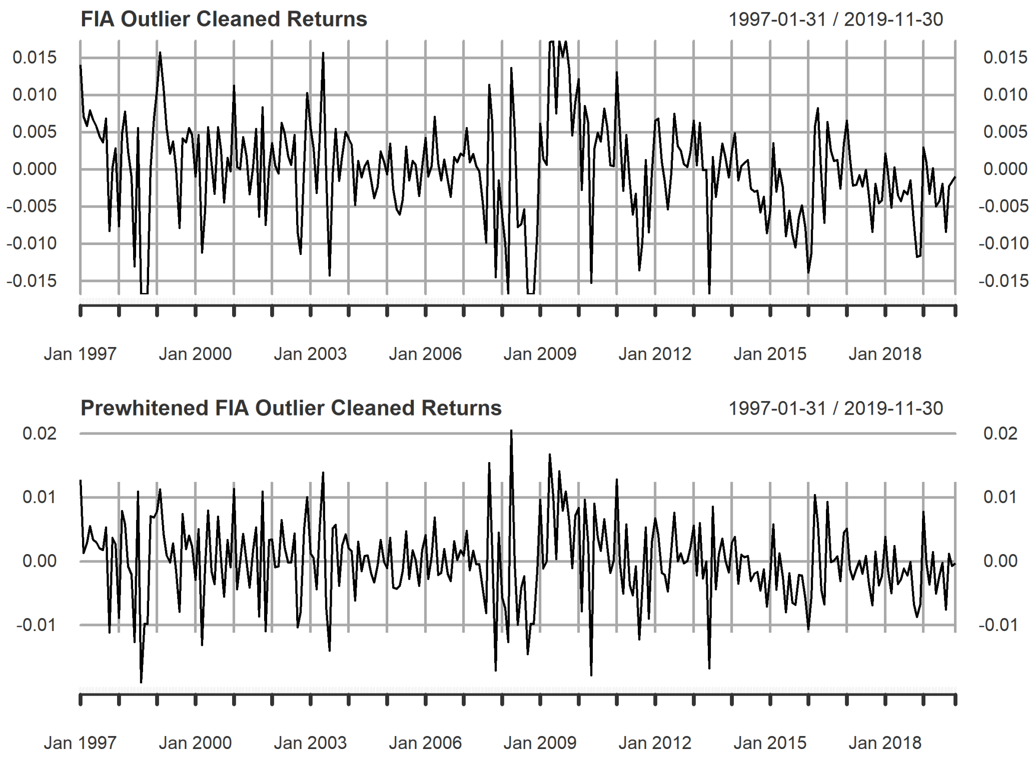


Figure 6: Outlier cleaned IF-transformed FIA returns and its prewhitened version.

For more information on all the features and visualization tools in the **RPEIF** package, see `https://CRAN.R-project.org/package=RPEIF`, where a reference manual and vignette are provided.

Periodogram Based GLM Method

The problem of estimating the variance (10) of a risk or performance estimator in the presence of serially correlated returns reduces to estimating the spectral density at zero frequency of the time series $IF(r_t; T, F)/n$. To do so, Chen and Martin (2021) proposed to fit a polynomial Gamma GLM method with elastic net (EN) regularization to the periodogram of the time series $IF(r_t; T, F)/n$. Because of the computationally intensive nature of the cross-validation procedure for the tuning parameter λ and the fact that the implementation must be efficient enough to handle multiple assets and portfolio returns, Chen et al. (2018) developed an accelerated proximal gradient descent algorithm for Gamma GLM with EN regularization.

The RPEGLMEN Package

The accelerated proximal gradient descent algorithm by Chen et al. (2018) was implemented in the **RPEGLMEN** package. Multicore processing is available in the penalized Gamma GLM functions within the package. The EN-penalized Exponential GLM is available as a special case of the Gamma GLM. For code examples and more details on the parallelization features in **RPEGLMEN**, see `https://CRAN.R-project.org/package=RPEGLMEN`, where a reference manual and vignette are provided.

Application: Hedge Funds Data Standard Errors with RPESE

The following functions are available in **RPESE** are:

```
library(RPESE)
ls("package:RPESE")

 [1] "DSR.SE"         "ES.SE"          "ESratio.SE"     "EstimatorSE"
 [5] "LPM.SE"         "Mean.SE"        "OmegaRatio.SE"  "printSE"
 [9] "RachevRatio.SE" "robMean.SE"     "SD.SE"          "SemiSD.SE"
 [13] "SoR.SE"         "SR.SE"          "VaR.SE"         "VaRratio.SE"
```

The only argument that is required for the standard error computing functions in RPESE from Table 1 is the data argument, and if only the data argument is supplied then the function uses the defaults of the other arguments. The arguments for the SD.SE and SR.SE functions given below show that the default GLM distribution is the exponential distribution for both the SD risk estimator and the SR performance estimator, but the se.methods defaults are the pair of IFiid and IFcor for the SD estimator and the pair IFiid and IFcorPW for the SR estimator.

```
args(SD.SE)

## function (data, se.method = c("IFiid", "IFcor", "IFcorAdapt",
##     "IFcorPW", "BOOTiid", "BOOTcor")[1:2], cleanOutliers = FALSE,
##     fitting.method = c("Exponential", "Gamma")[1], d.GLM.EN = 5,
##     freq.include = c("All", "Decimate", "Truncate")[1], freq.par = 0.5,
##     corOut = c("none", "retCor", "retIFCor", "retIFCorPW")[1],
##     return.coef = FALSE, ...)
```

```
args(SR.SE)

## function (data, rf = 0, se.method = c("IFiid", "IFcor", "IFcorAdapt",
##     "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 4)], cleanOutliers = FALSE,
##     fitting.method = c("Exponential", "Gamma")[1], d.GLM.EN = 5,
##     freq.include = c("All", "Decimate", "Truncate")[1], freq.par = 0.5,
##     corOut = c("none", "retCor", "retIFCor", "retIFCorPW")[1],
##     return.coef = FALSE, ...)
```

The standard error of the SD for the hedge funds in the edhec package can be computed, using the default arguments, with the following code:

```
SRout <- SR.SE(edhec)
```

The result returned by the function is a list. A more compact display of the results, with rounding to three digits by default, can be obtained using the printSE function, whose arguments are

```
args(printSE)

## function (SE.data, round.digit = 3, round.out = TRUE)
```

with the resulting output with this function for the SR is given below.

```
printSE(SRout)

##          SR IFiid IFcor IFcorPW
## CA    0.338 0.096 0.117   0.203
## CTAG  0.180 0.060 0.057   0.057
## DIS   0.392 0.080 0.108   0.134
## EM    0.194 0.069 0.084   0.092
## EMN   0.543 0.110 0.116   0.124
## ED    0.372 0.079 0.100   0.111
## FIA   0.377 0.113 0.134   0.183
## GM    0.371 0.054 0.057   0.057
## LS    0.315 0.066 0.079   0.085
## MA    0.560 0.092 0.098   0.104
## RV    0.504 0.097 0.117   0.169
## SS   -0.041 0.061 0.072   0.072
## FoF   0.275 0.066 0.083   0.093
```

Standard Errors Methods

The se.method argument is particularly important, and the standard error computation methods for the various choices for this argument are as follow.

- `"IFiid"`: This results in an influence function (IF) method based computation of a standard error assuming i.i.d. returns.
- `"IFcor"`: This is the basic IF method computation of a standard error that takes into account serial dependence in the returns when the correlation is not too large. This is often the best method for risk estimators.
- `"IFcorAdapt"`: This IF based method adaptively interpolates between IFcor and IFcorPW which is a good approach when the user is uncertain of the degree of serial dependence in the returns.
- `"IFcorPW"`: This IF based method uses pre-whitening of the IF-transformed returns and is useful when serial correlation is large.
- `"BOOTiid"`: This choice results in computing a bootstrap standard error assuming i.i.d. returns.
- `"BOOTcor"`: This choice uses a block bootstrap method to compute a standard error that takes into account serial dependence of returns.

The two default choices of methods are:

- `"IFiid"` and `"IFcor"` for risk estimators, and for performance estimators when returns serial correlation are known to be small, and
- `"IFiid"` and `"IFcorPW"` for performance estimators when returns correlations are unknown and may be large.

The following code results in standard errors for all thirteen of the `edhec` hedge funds, using five different methods methods.

```
SRout <- SR.SE(edhec, se.method = c("IFiid","BOOTiid","IFcor","IFcorPW","BOOTcor"))
printSE(SRout)

##          SR IFiid BOOTiid IFcor IFcorPW BOOTcor
## CA    0.338 0.096   0.101 0.117   0.203   0.146
## CTAG  0.180 0.060   0.062 0.057   0.057   0.039
## DIS   0.392 0.080   0.087 0.108   0.134   0.110
## EM    0.194 0.069   0.068 0.084   0.092   0.074
## EMN   0.543 0.110   0.098 0.116   0.124   0.184
## ED    0.372 0.079   0.076 0.100   0.111   0.084
## FIA   0.377 0.113   0.113 0.134   0.183   0.155
## GM    0.371 0.054   0.059 0.057   0.057   0.052
## LS    0.315 0.066   0.058 0.079   0.085   0.072
## MA    0.560 0.092   0.092 0.097   0.104   0.066
## RV    0.504 0.097   0.109 0.099   0.169   0.129
## SS   -0.041 0.061   0.065 0.072   0.072   0.076
## FoF   0.275 0.066   0.073 0.083   0.094   0.087
```

The value of including `IFiid`, along with `IFcor` and `IFcorPW` is that it allows the user to see whether or not serial correlation results in a difference in the standard error that assumes i.i.d. returns and the standard error that takes into account serial dependence. If there is no serial correlation there will not be much difference, but if there is serial correlation the difference can be considerable.

The `BOOTiid` and `BOOTcor` methods are provided for users who want to see how these bootstrap methods of computing standard errors compare with the IF based methods. Previous numerical experiments indicate that `BOOTiid` generally agrees quite well with `IFiid`, but that `BOOTcor` is not as consistent in giving values similar to those of `IFcor`.

Outliers Cleaning

The following code may be used to compare SR standard errors without and with outlier cleaning.

```
SRout <- SR.SE(edhec, se.method = "IFcorPW", cleanOutliers = F)
SRoutClean <- SR.SE(edhec, se.method = "IFcorPW", cleanOutliers = T)
clean.compare <- data.frame(SRout$IFcorPW$se, SRoutClean$IFcorPW$se)
names(clean.compare) <- c("With Outliers", "Outliers Cleaned")
row.names(clean.compare) <- names(edhec)
round(clean.compare, 3)

##      With Outliers Outliers Cleaned
## CA           0.203            0.116
## CTAG         0.057            0.057
## DIS          0.134           0.127
## EM           0.092           0.086
## EMN          0.124           0.099
## ED           0.113           0.101
## FIA          0.183           0.116
## GM           0.057           0.060
## LS           0.085           0.083
## MA           0.104           0.094
## RV           0.169           0.106
## SS           0.072           0.073
## FoF          0.093           0.084
```

Correlations of Returns and IF-transformed Returns

The (lag-1) correlations of the returns and IF-transformed returns time series can be computed as part of the output using the `corOut` argument. The options are `"retCor"`, `"retIFCor"` and `"retIFCorPW"`. Below is example code to return the correlations for the returns and the IF-transformed returns.

```
SRretCor <- SR.SE(edhec, corOut = c("retCor", "retIFCor"))
printSE(SRretCor)
```

```
##           SR IFiid IFcorPW retCor retIFCor
## CA     0.338 0.096   0.202  0.565    0.554
## CTAG   0.180 0.060   0.057 -0.008   -0.039
## DIS    0.392 0.080   0.134  0.492    0.486
## EM     0.194 0.069   0.092  0.296    0.280
## EMN    0.543 0.110   0.124  0.284    0.153
## ED     0.372 0.079   0.111  0.341    0.334
## FIA    0.377 0.113   0.183  0.500    0.440
## GM     0.371 0.054   0.057  0.057    0.060
## LS     0.315 0.066   0.086  0.216    0.252
## MA     0.560 0.092   0.104  0.277    0.130
## RV     0.504 0.097   0.169  0.424    0.480
## SS    -0.041 0.061   0.072  0.154    0.154
## FoF    0.275 0.066   0.093  0.311    0.334
```

Gamma and Exponential Distributions

In addition to the EN-penalized Gamma GLM available for the model-fitting step to the IF-transformed returns time series, the EN-penalized Exponential GLM is also available in **RPEGLMEN**. While the periodogram has an exponential distribution asymptotically, further research may show theat the Gamma distribution provides better overall results, particularly for non-normally distributed returns. The argument `fitting.method` specifies the GLM choice, and the `d.GLM.EN` argument specifies the polynomial degree for the model. By way of example, the following code computes standard errors of the SR for the exponential and Gamma distributions.

```
Clean.out <- SR.SE(edhec, se.method=c("IFiid","IFcor","IFcorPW"), cleanOutliers = T)
GammaClean.out <- SR.SE(edhec, se.method=c("IFiid","IFcor","IFcorPW"), cleanOutliers = T,
                        fitting.method = "Gamma")
GammaExp.comparison <- cbind(printSE(Clean.out)[,4], printSE(GammaClean.out)[,4])
colnames(GammaExp.comparison) <- c("IFcorPW-Exponential", "IFcorPW-Gamma")
rownames(GammaExp.comparison) <- names(edhec)
GammaExp.comparison
```

```
##      IFcorPW-Exponential IFcorPW-Gamma
## CA                 0.116         0.116
## CTAG               0.057         0.051
## DIS                0.127         0.128
## EM                 0.086         0.101
## EMN                0.099         0.114
## ED                 0.101         0.109
## FIA                0.116         0.120
## GM                 0.060         0.061
## LS                    0.083         0.081
## MA                    0.094         0.094
## RV                    0.106         0.098
## SS                    0.073         0.071
## FoF                   0.084         0.092
```

Decimation and Truncation of Frequencies of Discrete Fourier Transform

There is an option in the SE functions to use a decimated or truncated percentage of the frequencies of the discrete Fourier transforms for the periodogram in the fitting of the Gamma distributions. Decimation implies that only certain frequencies are used, and they will be equally spaced selections from the frequencies. Truncation implies that only a certain percentage of the frequencies (i.e. only the first frequencies until a certain point) will be used. By default, the standard error functions use all the frequencies. If the argument `freq.include` is set to "Decimate" or "Truncate" a value of 0.5 is used for the `freq.par` argument: every second frequency is used in the decimation case, and only the first half of the frequencies are used in the truncation case. Below is some sample code demonstration.

```
SRall <- SR.SE(edhec, cleanOutliers = T, freq.include = "All")
SRdecimate <- SR.SE(edhec, cleanOutliers = T, freq.include = "Decimate",
                    freq.par = 0.5)
SRtruncate <- SR.SE(edhec, cleanOutliers = T, freq.include = "Truncate",
                    freq.par = 0.5)
frequency <- cbind(printSE(SRall)[,3], printSE(SRdecimate)[,3],
                   printSE(SRtruncate)[,3])
colnames(frequency) <- c("IFcorPW-All", "IFcorPW-Decimate", "IFcorPW-Truncate")
rownames(frequency) <- names(edhec)
frequency
```

```
##      IFcorPW-All IFcorPW-Decimate IFcorPW-Truncate
## CA         0.116            0.121            0.113
## CTAG       0.057            0.057            0.057
## DIS        0.127            0.132            0.126
## EM         0.086            0.094            0.088
## EMN        0.099            0.109            0.102
## ED         0.101            0.105            0.100
## FIA        0.116            0.122            0.116
## GM         0.060            0.060            0.061
## LS         0.083            0.086            0.083
## MA         0.094            0.097            0.093
## RV         0.106            0.116            0.109
## SS         0.073            0.079            0.072
## FoF        0.084            0.087            0.088
```

Monte Carlo Study with IF-Based Standard Errors

To assess the accuracy of risk and performance estimators standard errors for the proposed method in comparison with alternative methods available for practitioners, a Monte Carlo simulation is carried out for the SR estimator. The standard error methods included in the simulation study are the IF-based method, the Newey-West (NW) HAC method (Newey and West, 1987) and the Andrews (AN) HAC method (Andrews, 1991), as well as their prewhitened versions using the `nse.andrews` and `nse.nw` functions in the **nse** package[3]. The Monte Carlo study compares the rejection probabilities of 95% confidence intervals based on the standard error estimates of the methods. The simulations are conducted using $N = 5{,}000$ replications of AR(1) processes with sample sizes $n = 120, 240$. The normal and t-distribution with $df = 5$ are considered for the innovations of the AR(1) processes, where the mean $\mu = 1\%$ and the volatility $\sigma = 15\%$. The simulation process is as follows:

1. A sample of size n is simulated from an AR(1) process, where the innovations follow either the normal or t-distribution ($df = 5$) with mean $\mu = 1\%$ and volatility $\sigma = 15\%$.
2. The SR is estimated using the sample mean and standard deviation, $\widehat{SR} = \frac{\hat{\mu}-rf}{\hat{\sigma}}$.
3. Steps 1. and 2. are repeated $N = 5{,}000$ times resulting in the estimates $\widehat{SR}_1, \widehat{SR}_2, \ldots, \widehat{SR}_N$. The "true" standard error of the SR is computed as the sample standard deviation of the N estimates.

[3]The **nse** package is a wrapper for the **sandwich** package.

4. For each of the N simulated time series of returns, the standard error of the SR is computed using the IF, HAC AN and NW methods, as well as for their prewhitened versions.
5. Based on a normal approximation for the SR, a nominal 5% error rate confidence interval is computed for each $\widehat{SR}_i, i = 1, 2, \ldots, N$, where the confidence interval is given by

$$\left(\widehat{SR}_i - t_{\alpha/2,n-1} \times SE_i, \widehat{SR}_i + t_{\alpha/2,n-1} \times SE_i\right),$$

and $t_{\alpha/2,n-1}$ is the $(1-\alpha/2)$-th quantile of the t-distribution with $n-1$ degrees of freedom. The rejection probabilities are computed as the fraction of the N replicates for which the replicate confidence interval does not contain the true SR.

Steps 1-6 of the Monte Carlo simulation study are repeated for each combination of $n = 60, 120, 240$ and $\phi = 0, 0.1, \ldots, 0.5$. The results for the normally and t-distributed innovations are provided in Tables 2 and 3, respectively, for a nominal error rate of 5%. For all of the methods, the prewhitening step improved the rejection probabilities. The IF-PW method was the best method overall, achieving rejection probabilities that were closest to 5% when averaging over all values of ϕ. In particular, the IF-PW method was also a highly competitive method even in the i.i.d. case ($\phi = 0$) for both the normally and t-distributed innovations, across all sample sizes considered. For a more extensive benchmark study of the new method and HAC-based methods, including a comparison of the methods under generalized autoregressive conditional heteroscedasticity (GARCH) models, see Chen and Martin (2021).

	AR(1) Parameter Values						
	$\phi = 0$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	**Avg. RP**
$n = 60$							
IF-PW	5.5	5.3	5.4	6.2	5.9	7.0	5.9
AN-PW	6.0	6.2	6.2	7.6	7.5	9.3	7.1
NW-PW	7.4	7.6	7.1	8.2	8.0	9.7	8.0
IF	5.5	5.5	5.8	7.2	8.0	11.6	7.3
AN	5.3	6.4	7.0	9.5	10.3	13.1	8.6
NW	8.4	9.0	8.9	10.1	10.9	13.7	10.2
$n = 120$							
IF-PW	5.2	4.6	5.0	6.0	5.5	5.8	5.3
AN-PW	5.5	4.8	5.4	6.5	6.7	7.3	6.0
NW-PW	6.5	5.9	6.2	7.0	7.1	7.8	6.7
IF	5.3	4.8	5.6	7.0	7.8	10.4	6.8
AN	5.2	5.5	7.0	8.4	8.8	10.2	7.5
NW	7.1	6.7	7.7	8.7	9.1	10.8	8.4
$n = 240$							
IF-PW	4.7	5.3	5.4	4.8	5.4	5.4	5.2
AN-PW	4.8	5.4	5.7	5.3	6.0	6.5	5.6
NW-PW	5.1	5.8	6.0	5.6	6.1	6.5	5.8
IF	4.8	5.5	5.9	5.8	7.8	10.4	6.7
AN	4.6	6.2	7.1	6.7	7.8	8.7	6.8
NW	5.4	6.3	7.2	7.0	7.9	8.8	7.1

Table 2: Rejection probabilities (RP) of the standard error methods for normally distributed innovations. The nominal error rate is 5%.

	AR(1) Parameter Values						
	$\phi = 0$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	Avg. RP
$n = 60$							
IF-PW	6.6	6.7	6.9	7.2	6.7	7.9	7.0
AN-PW	7.0	7.4	8.1	8.2	8.2	9.9	8.1
NW-PW	8.2	8.1	9.1	8.8	8.8	10.2	8.9
IF	6.7	6.8	7.5	8.6	8.6	12.0	8.4
AN	6.6	7.8	9.7	10.7	11.3	13.4	9.9
NW	9.5	9.7	10.9	11.4	11.8	14.0	11.2
$n = 120$							
IF-PW	6.2	5.7	6.2	5.9	6.9	5.8	6.1
AN-PW	6.5	5.9	6.8	6.4	7.8	7.1	6.7
NW-PW	7.7	7.1	7.6	7.3	8.4	7.4	7.6
IF	6.4	5.9	6.9	7.1	9.6	9.9	7.6
AN	6.0	6.7	8.3	8.5	10.8	10.2	8.4
NW	7.8	8.0	9.0	9.3	11.1	10.4	9.3
$n = 240$							
IF-PW	5.5	5.7	5.4	5.7	5.9	6.1	5.7
AN-PW	5.7	5.9	5.5	6.1	6.4	6.6	6.0
NW-PW	6.1	6.3	5.9	6.3	6.7	6.9	6.4
IF	5.6	5.9	5.9	6.8	8.2	10.2	7.1
AN	5.6	6.8	6.9	7.6	8.3	8.9	7.3
NW	6.4	7.0	6.6	7.7	8.4	8.9	7.5

Table 3: Rejection probabilities (RP) of the standard error methods for t-distributed innovations with $df = 5$. The nominal error rate is 5%.

Summary

This article introduced the **RPESE** package, as well as the related packages **RPEIF** and **RPEGLMEN**, to compute standard errors for risk and performance estimators using the new methodology in Chen and Martin (2021). The new methodology involves the representation of risk and performance estimators in terms of their IF-transformed returns, and fitting a polynomial Gamma GLM to the spectral density of the IF time series. The **RPEIF** package implements the IF computation for six risk estimators and nine performance estimators, and provides graphical visualization tools for the IFs. The **RPEGLMEN** implements an accelerated proximal gradient algorithm for the computation of the EN-penalized polynomial Gamma GLM applied to the spectral density of the IF-transformed returns, which includes multicore parallelization capabilities.

Code examples, real data applications and benchmark simulation studies in this article demonstrate the user-friendly software implementation of the method to assess the accuracy of risk and performance estimators, providing financial risk and portfolio managers with a powerful open-source toolbox. We note that the **PerformanceAnalytics** package uses the **RPESE** package to make its standard errors for serially dependent data available to the large base of quantitatiave finance users.

There is much further research that can be accomplished for the methodology of Chen and Martin (2021) and its software implementation in **RPESE**. New risk and performance estimators can be introduced to both **RPEIF** and **RPESE**. Alternative model selection methods could be applied in **RPESE** in addition to EN regularization, such as AIC or BIC based model selection. The proposed methodology could be applied to multivariate time series to study the joint behavior of risk and performance estimators. In the latter case, a surface fitting method will be required to apply to the multivariate spectral density of the multivariate IF-transformed returns.

Software and Data Availability

The package is available from the Comprehensive R Archive Network at https://CRAN.R-project.org/package=RPESE, where a reference manual and vignette are provided. The development website is available at https://github.com/AnthonyChristidis/RPESE. The scripts to replicate the code examples, the hedge fund data results and the Monte Carlo simulation are available at https://github.com/AnthonyChristidis/RPESE_RJournal_Simulation.

Acknowledgments

The software packages introduced in this article were developed while Anthony-Alexander Christidis was funded by a Google Summer of Code Award under the supervision of Dr. R. Doug Martin, Professor Emeritus of Applied Mathematics and Statistics at the University of Washington.

Bibliography

D. W. Andrews. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica: Journal of the Econometric Society*, pages 817–858, 1991. URL https://www.jstor.org/stable/2938229. [p697, 708]

X. Chen and R. D. Martin. Standard errors of risk and performance estimators for serially correlated returns. *Journal of Risk*, 23:1–41, 2021. URL https://doi.org/10.21314/JOR.2020.446. [p697, 699, 703, 704, 709, 710]

X. Chen, A. Y. Aravkin, and R. D. Martin. Generalized linear model for gamma distributed variables via elastic net regularization. 2018. URL https://arxiv.org/abs/1804.07780. [p704]

F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974. URL https://www.jstor.org/stable/2285666. [p698, 699]

F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, 1986. URL https://doi.org/10.1002/9781118186435. [p698]

P. Heidelberger and P. D. Welch. A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM*, 24(4):233–245, 1981. URL https://doi.org/10.1145/358598.358630. [p697]

H. R. Kunsch. The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, pages 1217–1241, 1989. URL https://www.jstor.org/stable/2241719. [p697]

O. Ledoit and M. Wolf. Robust performance hypothesis testing with the sharpe ratio. *Journal of Empirical Finance*, 15(5):850–859, 2008. URL https://doi.org/10.1016/j.jempfin.2008.03.002. [p697]

A. W. Lo. The statistics of sharpe ratios. *Financial Analysts Journal*, 58(4):36–52, 2002. URL https://doi.org/10.2469/faj.v58.n4.2453. [p697]

R. Maronna, R. Martin, V. Yohai, and M. Salibian-Barrera. *Robust Statistics: Theory and Methods (with R) Wiley*. 2019. URL https://doi.org/10.1002/9781119214656. [p703]

R. D. Martin and S. Zhang. Nonparametric versus parametric expected shortfall. *Journal of Risk*, 21(6): 1–41, 2019. URL https://ssrn.com/abstract=2747179. [p698]

W. K. Newey and K. D. West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703–708, 1987. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913610. [p697, 708]

D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994. URL https://doi.org/10.1080/01621459.1994.10476870. [p697]

A. Zeileis. Econometric computing with hc and hac covariance matrix estimators. *Journal of Statistical Software*, 2004. URL https://doi.org/10.18637/jss.v011.i10. [p697]

S. Zhang, R. D. Martin, and A.-A. Christidis. Influence functions for risk and performance estimators. *Journal of Mathematical Finance*, 11(1), 2021. URL https://doi.org/10.4236/jmf.2021.111002. [p700]

Anthony-Alexander Christidis
Department of Statistics
University of British Columbia
Vancouver, British Columbia, Canada
anthony.christidis@stat.ubc.ca

R. Doug Martin
Department of Applied Mathematics
University of Washington
Seattle, Washington, United States
doug@amath.washington.edu

RobustBF: An R Package for Robust Solution to the Behrens-Fisher Problem

by Gamze Güven, Şükrü Acıtaş, Hatice Şamkar and Birdal Şenoğlu

Abstract Welch's two-sample *t*-test based on least squares (LS) estimators is generally used to test the equality of two normal means when the variances are not equal. However, this test loses its power when the underlying distribution is not normal. In this paper, two different tests are proposed to test the equality of two long-tailed symmetric (LTS) means under heterogeneous variances. Adaptive modified maximum likelihood (AMML) estimators are used in developing the proposed tests since they are highly efficient under LTS distribution. An R package called **RobustBF** is given to show the implementation of these tests. Simulated Type I error rates and powers of the proposed tests are also given and compared with Welch's *t*-test based on LS estimators via an extensive Monte Carlo simulation study.

Introduction

Testing the equality of two population means is one of the most encountered problems in applied sciences. Student's *t*-test, which is uniformly most powerful unbiased, is commonly used under normality and homogeneity of variances assumptions. The well-known Behrens-Fisher (BF) problem arises when the assumption of homogeneity of variances is not met. This problem can be defined as testing the null hypothesis

$$H_0 : \mu_1 = \mu_2 \tag{1}$$

when $Y_{i1}, Y_{i2}, ..., Y_{in_i}$ $(i = 1, 2)$ are independent random samples from $N\left(\mu_i, \sigma_i^2\right)$ distribution. Fisher (1939) endorsed Behrens' solution to the BF problem by using the fiducial theory. Many researchers studied this problem. For example, Welch (1938) proposed a test statistic and provided its degrees of freedom approximately. It should be noted that degrees of freedom provided by Welch (1938) can also be obtained by using the Satterthwaite approximation; see Satterthwaite (1946). This is why the mentioned degrees of freedom is also known as Welch-Satterthwaite degrees of freedom in the literature. Wang (1971) calculated the Type I error rates of the Welch's two-sample *t*-test and Aspin-Welch test for different sets of degrees of freedom and nominal significance levels and concluded that Welch's *t*-test could be used in practice with little loss of accuracy. Davenport and Webster (1975) considered the test suggested by Fairfield Smith (1936) for the BF problem and compared its Type I error rates with those of Mehta and Srinivasan (1970). They concluded that this test is a very practical solution to the BF problem besides being stable in regard to size and having adequate power. Best and Rayner (1987) calculated the Wald score and likelihood ratio statistics and showed that the test based on Wald statistics has the same asymptotic properties as the Welch's *t*-test. Kim and Cohen (1998) presented a review of basic concepts and applications concerning the BF problem under fiducial, Bayesian and frequentist approaches. Singh et al. (2002) developed a test based on the Jackknife estimator of the common population variance and compared the powers of the proposed test with those of Welch's *t*-test and Cochran and Cox (1957) test. According to the results of their study, the proposed test is more powerful than the Cochran-Cox test for all cases, while it is more preferable to Welch's *t*-test for some cases. Chang and Pal (2008) developed a computational approach test (CAT) for the BF problem and compared it with Welch's *t*-test, Cochran-Cox text, Generalized *p*-value test, and Singh–Saxena–Srivastava test under the normal and *t*-model. They found that Welch's *t*-test, Cochran-Cox text, and CAT are robust under the heavier tailed *t*-models besides having similar size and power.

When the literature is examined, it can be seen that Welch's *t*-test has a very good performance as compared to other tests in the case of heteroscedasticity and unequal sample sizes in the context of normality. The power of Welch's *t*-test decreases very rapidly when the underlying distribution is long-tailed symmetric (LTS) since the least squares (LS) estimators are not robust to the violation of normality. It is known that non-normal distributions are more common in real-life problems. Yuen (1974) proposed a two-sample trimmed *t*-test and compared its performance of it with Welch's *t*-test for both normal and long-tailed samples. Tiku and Singh (1981) proposed Welch-type statistics based on modified maximum likelihood (MML) estimators and showed that the proposed test is more powerful than Yuen (1974)'s trimmed *t*-test. In addition, Tiku and Singh (1981) investigated the analogous test based on the robust bisquare estimators BS82 and showed that this test statistic gives misleading Type I errors.

In this study, a robust version of Welch's *t*-test for the BF problem is proposed when the underlying distribution is LTS. A second test using the fiducial model, which is a special case of a functional model given by Dawid and Stone (1982), is also proposed; see Fisher (1933, 1935) for more information about the fiducial approach. The reason for including a robust version of fiducial-based test into this study is to see its performance in the context of BF problem and to make comprehensive comparisons with its rivals (i.e., robust version of Welch's *t*-test and the traditional Welch's *t*-test). Both of the proposed tests are based on adaptive modified maximum likelihood (AMML) estimators, see Tiku and Sürücü (2009) and Dönmez (2010). To the best of our knowledge, this is the first study using AMML estimators for testing the equality of two LTS means under heterogeneous variances. These estimators are efficient and easy to compute for LTS samples, see Tiku and Sürücü (2009).

The R packages **stats** by R Core Team (1970) and **asht** by Fay (2020) include Welch's *t*-test based on LS estimators and BF test under normality, respectively. **WRS2** by Mair and Wilcox (2021) contains Yuen's test based on the trimmed sample means. Different from the earlier studies, we provide an R package **RobustBF** computing the values of the proposed test statistics and/or the corresponding *p*-values.

The rest of this study is organized as follows. Firstly, AMML estimators are given. Secondly, the robust Welch test and robust test based on the fiducial approach are developed. Thirdly, an extensive Monte Carlo simulation study is conducted to compare the performances of the proposed tests with the traditional Welch's *t*-test based on LS estimators. The proposed tests are applied to a real data set via **RobustBF** package. This paper is finalized some concluding remarks.

AMML Estimators

Assume that $Y_{i1}, Y_{i2}, ..., Y_{in_i}$ $(i = 1, 2)$ be independent random samples from $LTS\,(p, \mu_i, \sigma_i)$ distribution

$$f\,(y) = \frac{1}{\sqrt{k}\beta\,(1/2, p-1/2)\,\sigma}\left(1 + \frac{(y-\mu)^2}{k\sigma^2}\right)^{-p}, \quad -\infty < y < \infty; -\infty < \mu < \infty; \sigma > 0; p \geq 2, \tag{2}$$

where μ is the location parameter, σ is the scale parameter, p is the shape parameter, and $k = 2p - 3$ (Tiku and Kumra, 1985). It should be noted that $E\,(y) = \mu$, $V\,(y) = \sigma^2$, and $t = \sqrt{(\nu/k)}\,(y/\sigma)$ has Student's t distribution with $\nu = 2p - 1$ degrees of freedom.

The log-likelihood $(\ln L)$ function is given by

$$\ln L = -N \ln\left(\sqrt{k}\beta\,(1/2, p-1/2)\right) - \sum_{i=1}^{2} n_i \ln\,(\sigma_i) - p \sum_{i=1}^{2}\sum_{j=1}^{n_i} \ln\left(1 + \frac{\left(y_{ij} - \mu_i\right)^2}{k\sigma_i^2}\right), \tag{3}$$

where $N = n_1 + n_2$. Then, the likelihood equations are obtained as follows

$$\frac{\partial \ln L}{\partial \mu_i} = \frac{2p}{k\sigma_i}\sum_{j=1}^{n_i} g\left(z_{ij}\right) = 0 \tag{4}$$

$$\frac{\partial \ln L}{\partial \sigma_i} = -\frac{n_i}{\sigma_i} + \frac{2p}{k\sigma_i}\sum_{j=1}^{n_i} z_{ij}\, g\left(z_{ij}\right) = 0 \tag{5}$$

where

$$g\left(z_{ij}\right) = \frac{z_{ij}}{1 + (1/k)\, z_{ij}^2} \quad \text{and} \quad z_{ij} = \frac{y_{ij} - \mu_i}{\sigma_i}. \tag{6}$$

By solving the above likelihood equations, (4) and (5), simultaneously, the maximum likelihood (ML) estimators of the parameters μ_i and σ_i are obtained. However, these equations involve nonlinear functions of the parameters, and so ML estimators cannot be obtained explicitly. Hence, numerical methods can be used to solve these equations. Numerical methods may cause convergence problems like non-convergence of iterations, convergence to wrong roots, or multiple roots (Puthenpura and Sinha, 1986; Vaughan, 1992). MML methodology proposed by Tiku (1967, 1968) overcomes these mentioned problems by providing explicit solutions to likelihood equations. In MML methodology, firstly, the standardized statistics are ordered in ascending way, i.e., $z_{i(1)} \leq z_{i(2)} \leq ... \leq z_{i(n_i)}$. Then, likelihood equations in (4) and (5) are rewritten in terms of $z_{i(j)}$ and $g(z_{i(j)})$ $(i = 1, 2; j = 1, 2, ..., n_i)$ as shown in (7) and (8) since summation is invariant to ordering, i.e., $\sum_{j=1}^{n_i} z_{i(j)} = \sum_{j=1}^{n_i} z_{ij}$.

$$\frac{\partial \ln L}{\partial \mu_i} = \frac{2p}{k\sigma_i} \sum_{j=1}^{n_i} g\left(z_{i(j)}\right) = 0 \tag{7}$$

$$\frac{\partial \ln L}{\partial \sigma_i} = -\frac{n_i}{\sigma_i} + \frac{2p}{k\sigma_i} \sum_{j=1}^{n_i} z_{i(j)} g\left(z_{i(j)}\right) = 0. \tag{8}$$

Here, $z_{i(j)} = \frac{y_{i(j)}-\mu_i}{\sigma_i}$ and $g(z_{i(j)}) = \frac{z_{i(j)}}{1+(1/k)z_{i(j)}^2}$. The nonlinear function $g(z_{i(j)})$ is linearized utilizing the first two terms of the Taylor series expansion around the expected values of the ordered statistics $E(z_{i(j)}) = t_{i(j)}$ as follows

$$g\left(z_{i(j)}\right) \cong \alpha_{ij} + \beta_{ij} z_{i(j)}, \tag{9}$$

where

$$\alpha_{ij} = \frac{(2/k)\, t_{i(j)}^3}{\left(1 + (1/k)\, t_{i(j)}^2\right)^2} \quad \text{and} \quad \beta_{ij} = \frac{1 - (1/k)\, t_{i(j)}^2}{\left(1 + (1/k)\, t_{i(j)}^2\right)^2}. \tag{10}$$

Since $t_{i(j)}$ values cannot be obtained exactly, approximate values of $t_{i(j)}$ which do not affect the efficiencies of the resulting estimators are used,

$$\int_{-\infty}^{t_{i(j)}} f(z)\, dz = \frac{j}{n_i + 1}, \quad i = 1, 2; j = 1, 2, ..., n_i. \tag{11}$$

Secondly, modified likelihood equations are obtained by inserting the approximation (9) into Eqs. (7) and (8)

$$\frac{\partial \ln L^*}{\partial \mu_i} = \frac{2p}{k\sigma_i} \sum_{j=1}^{n_i} \left(\alpha_{ij} + \beta_{ij} z_{i(j)}\right) = 0 \tag{12}$$

$$\frac{\partial \ln L^*}{\partial \sigma_i} = -\frac{n_i}{\sigma_i} + \frac{2p}{k\sigma_i} \sum_{j=1}^{n_i} z_{i(j)} \left(\alpha_{ij} + \beta_{ij} z_{i(j)}\right) = 0. \tag{13}$$

Finally, MML estimators of μ_i and σ_i are found by solving Eqs. (12) and (13). They are given as follows

$$\hat{\mu}_i = \frac{\sum_{j=1}^{n_i} \beta_{ij} y_{i(j)}}{m_i} \quad \text{and} \quad \hat{\sigma}_i = \frac{B_i + \sqrt{B_i^2 + 4 n_i C_i}}{2\sqrt{n_i (n_i - 1)}}, \tag{14}$$

where

$$B_i = \frac{2p}{k} \sum_{j=1}^{n_i} \alpha_{ij} \left(y_{i(j)} - \hat{\mu}_i\right), \quad C_i = \frac{2p}{k} \sum_{j=1}^{n_i} \beta_{ij} \left(y_{i(j)} - \hat{\mu}_i\right)^2 \quad \text{and} \quad m_i = \sum_{j=1}^{n_i} \beta_{ij}; \tag{15}$$

see Tiku and Suresh (1992). The asymptotic properties of the MML estimators $\hat{\mu}_i$ and $\hat{\sigma}_i$ can be demonstrated with the help of the following theorems.

Theorem 1 *$\hat{\mu}_i$ is the minimum variance bound (MVB) estimator and is asymptotically normally distributed with mean μ_i and variance σ_i^2 / M_i $(M_i = 2pm_i/k)$.*

Theorem 2 *$(n_i - 1)\, \hat{\sigma}_i^2 / \sigma_i^2$ is distributed as chi-square (more accurately a multiple of chi-square) with $(n_i - 1)$ degrees of freedom.*

For proofs of theorems, see, e.g. Şenoğlu and Tiku (2001); Güven et al. (2019).

MML estimators have the same asymptotic properties as the ML estimators and are as efficient as ML estimators, even for small samples. They are easy to compute and robust to the outliers.

It should be noted that the shape parameter p is assumed to be known in the MML methodology. However, in some real-life applications, it may be possible to assume that the data comes from a certain type of distribution, namely LTS distribution, but there is no opportunity to specify the value of the shape parameter. Hence, Tiku and Sürücü (2009) proposed AMML methodology, which is a new version of MML methodology, see Dönmez (2010) and Acıtaş et al. (2020, 2021). This methodology relaxes the assumption of the known shape parameter. AMML estimators are computed in two iterations. In the first iteration, initial t_{ij} values are calculated from the sample data, as shown below

$$t_{ij} = \left(y_{ij} - T_{0i}\right) / S_{0i} \quad i = 1, 2; j = 1, ..., n_i. \tag{16}$$

Here, T_{0i} and S_{0i} are the initial estimates of μ_i and σ_i and they are calculated as

$$T_{0i} = med\left\{y_{ij}\right\} \quad \text{and} \quad S_{0i} = 1.483 med\left\{\mid y_{ij} - T_{0i} \mid\right\} \quad i = 1, 2; j = 1, ..., n_i, \tag{17}$$

respectively. Using the t_{ij} values in (16), α_{ij} and β_{ij} coefficients are calculated as follows

$$\alpha_{ij} = \frac{(1/k)\, t_{ij}}{1 + (1/k)\, t_{ij}^2} \quad \text{and} \quad \beta_{ij} = \frac{1}{1 + (1/k)\, t_{ij}^2}. \tag{18}$$

Then, the AMML estimates of the parameters μ_i and σ_i are obtained using Eq. (14) and α_{ij} and β_{ij} values given in Eq. (18). To distinguish these estimates from the MML estimates, they are represented by $\hat{\mu}_{i(AMML)}$ and $\hat{\sigma}_{i(AMML)}$ in the rest of the paper. In the second iteration, t_{ij} values are revised as follows

$$t_{ij} = \left(y_{ij} - \hat{\mu}_{i(AMML)}\right) / \hat{\sigma}_{i(AMML)} \quad i = 1, 2; j = 1, ..., n_i \tag{19}$$

and recalculate the α_{ij} and β_{ij} values using the equalities in (18) for these t_{ij} values. Then final AMML estimates of μ_i and σ_i are obtained.

It should be noted that in AMML methodology, y_{ij} observations are used rather than the ordered $y_{i(j)}$ observations since t_{ij} values are calculated from the sample observations. In addition, the shape parameter p is taken to be 16.5 in the calculations of α_{ij} and β_{ij} coefficients since this value makes AMML estimators efficient for normal and near normal distributions. It also makes them robust to mild outliers. The reason why we use AMML methodology in the proposed tests is that it provides the same asymptotic properties as MML methodology and, as mentioned before, relaxes the assumption of known shape parameter p.

Proposed Test Statistics

In this section, we propose two different tests for testing the equality of two LTS means.

Robust Welch (RW) Test

In this subsection, we briefly introduce Welch's t-test proposed by Welch (1938) under normal theory and then give the robust version of it. Welch's t-test based on LS estimators is defined as $W = \{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)\} / \sqrt{\{(s_1^2/n_1) + (s_2^2/n_2)\}}$. It is known that W is approximately distributed as Student's t with degrees of freedom

$$f = \frac{1}{\{c^2/\,(n_1 - 1) + (1 - c^2)\,/\,(n_2 - 1)\}}, \tag{20}$$

where $c = \left(s_1^2/n_1\right) / \left\{\left(s_1^2/n_1\right) + \left(s_2^2/n_2\right)\right\}$. Here, $\bar{x}_i$ and s_i^2 $(i = 1, 2)$ are the sample means and sample variances, respectively. The value of W test can be obtained using `t.test` function available in R.

In this study, we propose the following test statistics based on AMML estimators as a robust alternative to Welch's t-test

$$RW = \frac{\left(\hat{\mu}_{1(AMML)} - \hat{\mu}_{2(AMML)}\right) - (\mu_1 - \mu_2)}{\sqrt{\left(\hat{\sigma}_{1(AMML)}^2/M_1\right) + \left(\hat{\sigma}_{2(AMML)}^2/M_2\right)}}. \tag{21}$$

As we shall see at the end of this section, the null distribution of RW is approximately distributed as Student's t based upon Theorems 1 and 2 . The approximate degrees of freedom for this test is obtained using the Satterthwaite (1946) approximation as follows.

Let

$$c_1 = \frac{\sigma_1^2}{(n_1-1)\,M_1}, \qquad c_2 = \frac{\sigma_2^2}{(n_2-1)\,M_2} \tag{22}$$

$$Q_1 = \frac{(n_1-1)\,\hat{\sigma}^2_{1(AMML)}}{\sigma_1^2} \quad \text{and} \quad Q_2 = \frac{(n_2-1)\hat{\sigma}^2_{2(AMML)}}{\sigma_2^2} \tag{23}$$

where Q_1 and Q_2 are independent chi-square random variables with degrees of freedom (n_1-1) and (n_2-1), respectively (see Theorem 2). If the linear combination of Q_1 and Q_2 is written as

$$Q = c_1Q_1 + c_2Q_2 = \frac{\hat{\sigma}^2_{1(AMML)}}{M_1} + \frac{\hat{\sigma}^2_{2(AMML)}}{M_2}, \tag{24}$$

then $\nu Q/E(Q)$ has an approximate χ^2 distribution with the following degrees of freedom

$$\begin{aligned}\nu &= \frac{\left[c_1Q_1 + c_2Q_2\right]^2}{\left(\left[c_1Q_1\right]^2/\nu_1\right) + \left(\left[c_2Q_2\right]^2/\nu_2\right)} \\ &= \frac{\left(\left(\hat{\sigma}^2_{1(AMML)}/M_1\right) + \left(\hat{\sigma}^2_{2(AMML)}/M_2\right)\right)^2}{\left(\hat{\sigma}^2_{1(AMML)}/M_1\right)^2/(n_1-1) + \left(\hat{\sigma}^2_{2(AMML)}/M_2\right)^2/(n_2-1)}.\end{aligned} \tag{25}$$

Here,

$$\nu_1 = n_1 - 1, \quad \nu_2 = n_2 - 1 \quad \text{and} \quad E(Q) = \frac{\sigma_1^2}{M_1} + \frac{\sigma_2^2}{M_2}. \tag{26}$$

RW in (21) can be rewritten as follows

$$RW = \frac{\left(\left(\hat{\mu}_{1(AMML)} - \hat{\mu}_{2(AMML)}\right) - (\mu_1-\mu_2)\right)/\sqrt{(\sigma_1^2/M_1) + (\sigma_2^2/M_2)}}{\sqrt{\left(\hat{\sigma}^2_{1(AMML)}/M_1\right) + \left(\hat{\sigma}^2_{2(AMML)}/M_2\right)}/\sqrt{(\sigma_1^2/M_1) + (\sigma_2^2/M_2)}}. \tag{27}$$

Since this expression is equivalent to

$$\frac{Z}{\sqrt{Q}/\sqrt{E(Q)}}, \tag{28}$$

it is obvious that *RW* is approximately distributed as Student's t with ν degrees of freedom. Here, $Z \sim N(0,1)$ (see Theorem 1) and $\sqrt{Q}/\sqrt{E(Q)} \sim \sqrt{\chi^2_\nu/\nu}$.

To verify the null distribution of the *RW*, the probabilities

$$p_1 = Pr\left(|RW| \geq t_{1-\alpha/2,\nu}\right) \tag{29}$$

are simulated from 10,000 Monte Carlo runs for various combinations of the sample sizes n_1 and n_2. The results are demonstrated in Table 1. Here, ν is the degrees of freedom for *RW*.

Robust Fiducial (RF) Based Test

In this section, fiducial-based test is proposed using the concept of fiducial inference and pivotal model; see Fisher (1933, 1935) and Dawid and Stone (1982). Let denote the *RW* test based on the observed values as

$$RW^* = \frac{\left(\hat{\mu}^*_{1(AMML)} - \hat{\mu}^*_{2(AMML)}\right) - (\mu_1-\mu_2)}{\sqrt{\frac{\hat{\sigma}^{2*}_{1(AMML)}}{M_1} + \frac{\hat{\sigma}^{2*}_{2(AMML)}}{M_2}}}. \tag{30}$$

First, the fiducial distribution of RW^* is derived using pivotal quantities and fiducial distribution of the parameters of interest. Then, the corresponding p-value is obtained. Here, $\left(\hat{\mu}^*_{i(AMML)}, \hat{\sigma}^{2*}_{i(AMML)}\right)$

are the observed values of $\left(\hat{\mu}_{i(AMML)}, \hat{\sigma}^2_{i(AMML)}\right)$ $(i = 1, 2)$. Let

$$Z_i = \frac{\hat{\mu}_{i(AMML)} - \mu_i}{\sigma_i / \sqrt{M_i}} \tag{31}$$

and

$$Q_i = \frac{(n_i - 1)\, \hat{\sigma}^2_{i(AMML)}}{\sigma_i^2} \tag{32}$$

are mutually independent pivotal quantities. They have asymptotically $N(0,1)$ and $\chi^2_{(n_i-1)}$ distributions, respectively (see Theorems 1 and 2). Using pivotal quantities Z_i and Q_i, data generating equations are obtained as given below

$$\hat{\mu}_{i(AMML)} = \mu_i + \left(\sigma_i / \sqrt{M_i}\right) Z_i \tag{33}$$

and

$$\hat{\sigma}^2_{i(AMML)} = \sigma_i^2 Q_i / (n_i - 1)\,. \tag{34}$$

Given $\left(\hat{\mu}^*_{i(AMML)}, \hat{\sigma}^{2*}_{i(AMML)}\right)$, Eqs. (33) and (34) are expressed as follows

$$\hat{\mu}^*_{i(AMML)} = \mu_i + \left(\sigma_i / \sqrt{M_i}\right) z_i \tag{35}$$

and

$$\hat{\sigma}^{2*}_{i(AMML)} = \sigma_i^2 q_i / (n_i - 1)\,. \tag{36}$$

Here, (z_i, q_i) are the observed values of (Z_i, Q_i). Eqs. (35) and (36) have the unique solutions as given below

$$\mu_i = \hat{\mu}^*_{i(AMML)} - \frac{z_i}{\sqrt{q_i / (n_i - 1)}} \frac{\hat{\sigma}^*_{i(AMML)}}{\sqrt{M_i}} \tag{37}$$

and

$$\sigma_i^2 = \frac{(n_i - 1)\, \hat{\sigma}^{2*}_{i(AMML)}}{q_i}. \tag{38}$$

Since $\frac{Z_i}{\sqrt{Q_i/(n_i-1)}}$ is distributed as a t_i variable with $(n_i - 1)$ degrees of freedom, the fiducial distribution of μ_i is the same as that of

$$T^*_{\mu_i} = \hat{\mu}^*_{i(AMML)} - \frac{t_i \hat{\sigma}^*_{i(AMML)}}{\sqrt{M_i}} \tag{39}$$

for given $\left(\hat{\mu}^*_{(AMML)}, \hat{\sigma}^{2*}_{(AMML)}\right)$. Therefore, the fiducial distribution of RW^* in (30) is derived by utilizing the fiducial distribution of μ_i as follows

$$T_{RF} = \frac{\left(\left(t_1 \hat{\sigma}^*_{1(AMML)}\right) / \sqrt{M_1}\right) - \left(\left(t_2 \hat{\sigma}^*_{2(AMML)}\right) / \sqrt{M_2}\right)}{\sqrt{\left(\hat{\sigma}^{2*}_{1(AMML)}\right) / M_1 + \left(\hat{\sigma}^{2*}_{2(AMML)}\right) / M_2}}, \tag{40}$$

where $t_1 \sim t_{(n_1-1)}$ and $t_2 \sim t_{(n_2-1)}$. Since

$$RW_0^* = \frac{\left(\hat{\mu}^*_{1(AMML)} - \hat{\mu}^*_{2(AMML)}\right)}{\sqrt{\left(\hat{\sigma}^{2*}_{1(AMML)}\right) / M_1 + \left(\hat{\sigma}^{2*}_{2(AMML)}\right) / M_2}} \tag{41}$$

is the observed value of T_{RF} under $H_0 : \mu_1 = \mu_2$, the corresponding p-value is given by

$$p = Pr\left(T_{RF} \geq RW_0^*\right). \tag{42}$$

An algorithm for calculating the fiducial p-value in Eq.(42) via Monte Carlo simulation study is given as follows

Algorithm 1

```
Step 1 For the given data, compute $\hat{\mu}^*_{i(AMML)}$, $\hat{\sigma}^*_{i(AMML)}$ $(i = 1, 2)$ and then $RW^*_0$ utilizing Eq. (41).
Step 2 Generate $t_i \sim t(n_i - 1), (i = 1, 2)$.
Step 3 Compute $T^2_{RF}$ utilizing Eq. (40).
Step 4 Let $F_l = 1$ if $T^2_{RF} > RW^{2*}_0$, else $F_l = 0$
Step 5 Repeat the steps 2-4 $K$ times.
Step 6 Compute the simulated $p$-value using $p = \frac{1}{K}\sum_{j=1}^{K} F_j$.
```

It should be noted that the squares of T_{RF} and RW^*_0 in Steps 3 and 4 are taken since the alternative hypothesis is two-sided, i.e., $H_1 : \mu_1 - \mu_2 \neq 0$; see Li et al. (2011).

Monte Carlo Simulation

In this section, Type I error rates and powers of the proposed tests (RW and RF) are compared with those of the W test under the specified nominal level $\alpha = 0.05$. The plan of the simulation study is outlined as follows:

We use the following population distributions while generating samples.

	Population 1	Population 2
(a)	$Cauchy(0,1)$	$Cauchy(0,1)$
(b)	$5 \times Cauchy(0,1)$	$Cauchy(0,1)$
(c)	$Normal(0,3^2)/Uniform(0,1)$	$Normal(0,1)/Uniform(0,1)$
(d)	$0.8Normal(0,4^2) + 0.2\frac{Normal(0,4^2)}{Uniform[0,1]}$	$0.8Normal(0,1) + 0.2\frac{Normal(0,1)}{Uniform[0,1]}$
(e)	$3t_2$	t_2
(f)	$2t_5$	t_5
(g)	$Logistic(0,3)$	$Logistic(0,1)$
(h)	$Laplace(0,1)$	$Laplace(0,\sqrt{6})$

Here, t_a: Student's t distribution with a degrees of freedom.

10,000 different samples are considered for each of size n_i $(i = 1, 2)$. Sample sizes are taken as (n_1, n_2)=$(6,6)$,$(6,10)$,$(10,10)$,$(10,15)$,$(10,30)$, $(20,20)$,$(20,30)$,$(20,50)$,$(30,50)$ and $(50,50)$ while comparing the Type I error rates and powers of the tests. Simulations are conducted in R software.

To compute the Type I error rates of the RW, RF, and W tests, firstly, samples are generated under the null hypothesis H_0:$\mu_1 = \mu_2$ for given (n_1, n_2) . Then AMML and LS estimates of the parameters are calculated. The probability in Eq. (29) gives the Type I rates of the RW test. It should be noted that this probability shows that how close the distribution of the RW test is to Student's t with degrees of freedom ν. RF is carried out using Algorithm 1 with K=5,000. The fiducial p-value for the RF is computed in the final step of the mentioned algorithm. This procedure is repeated for each of the 10,000 samples. The proportion of the 10,000 p-values that are less than the nominal level $\alpha = 0.05$ gives Type I error rates of the RF.

To compute the power of the tests, similar steps are followed, but a constant d is added to the observations in the first population. Any test can be considered powerful if it achieves maximum power and adheres to the prescribed significance level.

Results

The results of the Monte Carlo simulation study are given in Tables 1-9. The Type I error rates and power of the tests are given in Table 1 and Tables 2-9, respectively.

Numerical results of Table 1 can be summarized for Models (a)-(h) as follows

- Models (a)-(c): Type I error rates of the RW, RF, and W tests are smaller than the nominal level of $\alpha = 0.05$. However, Type I error rates of the RW test are much closer to the nominal level than those of RF and W. RW test is followed by RF. W test is conservative regardless of the sample sizes.
- Models (d)-(e): Type I error rates of the RW test are closest to the nominal level of $\alpha = 0.05$ for all sample sizes even for small ones. RW test is followed by RF. W test is conservative compared to the RW and RF.
- Models (f)-(h): Type I error rates of the RW, RF, and W tests are close to the nominal level of $\alpha = 0.05$. However, RW has very good performance in terms of Type I error rates, even for small samples. Type I error rates of the RF test are slightly smaller than the nominal level for small sample sizes in Model (h).

The numerical results of Tables 2-9 can be summarized as follows. It should be noted that the first line of Tables 2-9, that is, $d = 0.00$ presents simulated Type I error rates of the tests.

- Model (a): The RW test appears to be more powerful than the RF for small to moderate sample sizes. However, as sample sizes increase, powers of RW and RF tests get closer to each other. These two tests outperform the W test for all sample sizes. The power of the W test is decreasing with increasing sample sizes.
- Models (b)-(e): The RW and RF tests exhibit similar power properties, and they have the most power for all sample sizes. The W test has the least power. However, the W test shows the worst performance in Model (b) when sample sizes are large and equal, i.e., (n_1, n_2)=$(50, 50)$.
- Models (f)-(g): RW, RF, and W tests have similar power properties for small to moderate sample sizes. However, as sample sizes increase the RW and RF tests exhibit better performance than the W test.
- Model (h): The RW test has the most power followed by RF. The W test has the least power especially for moderate to large sample sizes.

Overall, the RW test can be recommended for testing the equality of two LTS means under the assumption of heterogeneous variances since it has the best performance with respect to size and power. Although the performance of the RF test is not as good as the RW test, it has better performance than the traditional Welch's t-test.

Using RobustBF package

In the **RobustBF** package, we show the implementation of the proposed tests (RW and RF), based on AMML estimators, and W test, based on LS estimators, using the data representing the values of $10(y - 2.0)$ (y is the pollution level (measurement of lead) in water samples from two lakes). It has been shown that long-tailed symmetric distribution provides a plausible model for the mentioned data; see Tiku and Akkaya (2004) and also reference therein.

To run **RobustBF** package, we first install the package and then load it by typing:

```
> install.packages("RobustBF")
> library(RobustBF)
```

respectively. Next the pollution level data are inputted for each lakes (Lake 1 and Lake 2) in terms of the vectors as shown below

```
y1 <- c(-1.48, 1.25, -0.51, 0.46, 0.60, -4.27, 0.63, -0.14, -0.38, 1.28,
         0.93, 0.51, 1.11, -0.17, -0.79, -1.02, -0.91, 0.10, 0.41, 1.11)
y2 <- c(1.32, 1.81, -0.54, 2.68, 2.27, 2.70, 0.78, -4.62, 1.88, 0.86,
        2.86, 0.47, -0.42, 0.16, 0.69, 0.78, 1.72, 1.57, 2.14, 1.62)
```

The value of the RW test, its degrees of freedom with the corresponding p-value, AMML estimates of the location parameters ($\hat{\mu}_{1(AMML)}$, $\hat{\mu}_{2(AMML)}$), and AMML estimates of the scale parameters ($\hat{\sigma}_{1(AMML)}$, $\hat{\sigma}_{2(AMML)}$) are given by using the function

```
> RW(y1,y2)
```

The p-value and AMML estimates of the location and scale parameters are given for the RF test by using the function

```
> RF(y1,y2,iter=5000)
```

It should be noted that the p-value for the RF test is obtained using a computational approach, and it is based on the replication number in Algorithm 1, denoted as `iter` in the `RF` function. When the above-mentioned functions in the **RobustBF** package are performed, the following results are obtained

```
> RW(y1,y2)

           Robust Welch's Two Sample t-Test

data:  y1 and y2
RW = -3.1602, df = 36.892, p-value = 0.0031
alternative hypothesis: true difference between in means is not equal to 0
sample estimates:
  mean of y1  mean of y2   sd of y1   sd of y2
      0.0626      1.2391     1.0861     1.2876

> RF(y1,y2,iter=5000)

          Robust Fiducial Based Test

data:  y1 and y2
p-value = 0.0032
alternative hypothesis: true difference in means is not equal to 0
sample estimates:
mean of y1 mean of y2   sd of y1   sd of y2
    0.0626     1.2391     1.0861     1.2876
```

We also use `t.test` function in R to test the null hypothesis H_0:$\mu_1 = \mu_2$ and obtain its p-value as 0.0243. It can be seen from these results, RW, RF, and W tests reject the null hypothesis at $\alpha = 0.05$ significance level since the p-values corresponding to these tests are all less than 0.05. However, p-values for RW and RF tests are much smaller than the ones obtained for W. Results of the RW and RF tests are more reliable since the AMML estimates of the σ_1 and σ_2 ($\hat{\sigma}_{1(AMML)} = 1.0861$, $\hat{\sigma}_{2(AMML)} = 1.2876$) are less than the corresponding LS estimates ($\hat{\sigma}_{1(LS)} = 1.2819$, $\hat{\sigma}_{2(LS)} = 1.6542$). It should be noted that RW and RF tests reject the null hypothesis while W fails to reject it at the significance level $\alpha = 0.01$. These results are in agreement with the simulation results in the context of long-tailed symmetric distributions.

Conclusion

Reviewing the literature shows that comparing two means is a commonly encountered problem, especially in applied sciences when the usual normality and homogeneity of variances assumptions are violated. For this reason, in this study, we present **RobustBF** package and propose RW and RF tests to test the equality of two LTS means when the variances are unknown and arbitrary. The first test included in the package is a robust version of Welch's t-test, and the other one is a robust fiducial-based test. The proposed tests are based on AMML estimators. Also, we use `t.test` function available in R to compare the proposed tests with Welch's t-test in terms of Type I error rates and powers. Examining the results of the simulation study reveals that Type I error rates of the RW test are closer to the nominal level in general. Therefore, the RW test verifies the obtained null distribution for long-tailed symmetric samples. This test is followed by RF. RF does not require the knowledge of sampling distribution of the test statistics. W test appears to be conservative except for the t_5, Logistic and Laplace distributions. RW shows the best power performance among the others besides being robust for the contamination model for the scenarios considered in this study. Therefore, the proposed RW test can be recommended for testing the equality of two LTS means under heterogeneity of variances. W test performs poorly in almost all cases. According to our knowledge, the proposed tests presented in the **RobustBF** package are not available in any other R tool.

Bibliography

S. Acıtaş, P. Filzmoser, and B. Şenoğlu. A new partial robust adaptive modified maximum likelihood estimator. *Chemometrics and Intelligent Laboratory Systems*, 204:104068, 2020. URL https://doi.org/10.1016/j.chemolab.2020.104068. [p716]

S. Acıtaş, P. Filzmoser, and B. Şenoğlu. A robust adaptive modified maximum likelihood estimator for the linear regression model. *Journal of Statistical Computation and Simulation*, 91(7):1394–1414, 2021. URL https://doi.org/10.1080/00949655.2020.1856847. [p716]

D. J. Best and J. C. W. Rayner. Welch's approximate solution for the behrens − fisher problem. *Technometrics*, 29(2):205–210, 1987. URL https://doi.org/10.1080/00401706.1987.10488211. [p713]

C. H. Chang and N. Pal. A revisit to the behrens–fisher problem: comparison of five test methods. *Communications in Statistics—Simulation and Computation*, 37(6):1064–1085, 2008. URL https://doi.org/10.1080/03610910802049599. [p713]

W. G. Cochran and G. M. Cox. *Experimental designs*. John Wiley and Sons, New York, 1957. [p713]

J. M. Davenport and J. T. Webster. The behrens-fisher problem, an old solution revisited. *Metrika*, 22: 47–54, 1975. URL https://doi.org/10.1007/BF01899713. [p713]

A. P. Dawid and M. Stone. The functional-model basis of fiducial inference. *The Annals of Statistics*, 10 (4):1054–1067, 1982. URL https://www.jstor.org/stable/2240708. [p714, 717]

A. Dönmez. *Adaptive estimation and hypothesis testing methods [dissertation]*. Ankara:METU, 2010. [p714, 716]

H. Fairfield Smith. The problem of comparing the result of two experiments with unequal errors. *Journal Council for Scientific and Industrial Research Australia*, 9:211–212, 1936. URL https://publications.csiro.au/publications/publication/PIprocite:a35f0b77-57ff-496e-869f-10ba39834723. [p713]

M. P. Fay. *asht: Applied Statistical Hypothesis Tests*, 2020. URL https://cran.r-project.org/web/packages/asht/index.html. R package version 0.9.6. [p714]

R. A. Fisher. The concepts of inverse probability and fiducial probability referring to unknown parameters. *Proceedings of the Royal Society of London. Series A*, 139(838):343–348, 1933. URL https://doi.org/10.1098/rspa.1933.0021. [p714, 717]

R. A. Fisher. The fiducial argument in statistical inference. *Annals of eugenics*, 6(4):391–398, 1935. URL https://doi.org/10.1111/j.1469-1809.1935.tb02120.x. [p714, 717]

R. A. Fisher. The comparison of samples with possibly unequal variances. *Annals of Eugenics*, 9(2): 174–180, 1939. URL https://doi.org/10.1111/j.1469-1809.1939.tb02205.x. [p713]

G. Güven, O. Gürer, H. Şamkar, and B. Şenoğlu. A fiducial-based approach to the one-way anova in the presence of nonnormality and heterogeneous error variances. *Journal of Statistical Computation and Simulation*, 89(9):1715–1729, 2019. URL https://doi.org/10.1080/00949655.2019.1593985. [p715]

S. H. Kim and A. S. Cohen. On the behrens-fisher problem: A review. *Journal of Educational and Behavioral Statistics*, 23(4):356–377, 1998. URL https://doi.org/10.3102/10769986023004356. [p713]

X. Li, J. Wang, and H. Liang. Comparison of several means: A fiducial based approach. *Computational statistics & data analysis*, 55(5):1993–2002, 2011. URL https://doi.org/10.1016/j.csda.2010.12.009. [p719]

P. Mair and R. Wilcox. *WRS2: A Collection of Robust Statistical Methods*, 2021. URL https://cran.r-project.org/web/packages/WRS2/index.html. R package version 1.1−1. [p714]

J. S. Mehta and R. Srinivasan. On the behrens-fisher problem. *Biometrika*, 57(3):649–655, 1970. URL https://doi.org/10.1093/biomet/57.3.649. [p713]

S. Puthenpura and N. K. Sinha. Modified maximum likelihood method for the robust estimation of system parameters from very noisy data. *Automatica*, 22(2):231–235, 1986. URL https://doi.org/10.1016/0005-1098(86)90085-3. [p714]

R Core Team. *The R Stats Package*, 1970. URL https://www.rdocumentation.org/packages/stats/versions/3.6.2. R package version 3.6.2. [p714]

F. E. Satterthwaite. An approximate distribution of estimates of variance components. *Biometrics Bulletin*, 2(6):110–114, 1946. URL https://doi.org/10.2307/3002019. [p713, 716]

B. Şenoğlu and M. L. Tiku. Analysis of variance in experimental design with nonnormal error distributions. *Communications in Statistics-Theory and Methods*, 30(7):1335–1352, 2001. URL https://doi.org/10.1081/STA-100104748. [p715]

P. Singh, K. K. Saxena, and O. P. Srivastava. Power comparisons of solutions to the behrens-fisher problem. *American Journal of Mathematical and Management Sciences*, 22(3):233–250, 2002. URL https://doi.org/10.1080/01966324.2002.10737589. [p713]

M. L. Tiku. Estimating the mean and standard deviation from a censored normal sample. *Biometrika*, 54:155–165, 1967. URL https://doi.org/10.1093/biomet/54.1-2.155. [p714]

M. L. Tiku. Estimating the parameters of log-normal distribution from censored samples. *Journal of the American Statistical Association*, 63(321):134–140, 1968. URL https://doi.org/10.2307/2283834. [p714]

M. L. Tiku and A. D. Akkaya. *Robust estimation and hypothesis testing*. New Age International Limited Publishers, New Delhi, 2004. [p720]

M. L. Tiku and S. Kumra. Expected values and variances and covariances of order statistics for a family of symmetric distributions (student's t). *Selected tables in mathematical statistics*, 8:141–270, 1985. [p714]

M. L. Tiku and M. Singh. Robust test for means when population variances are unequal. *Communications in Statistics—Theory and Methods*, 10(20):2057–2071, 1981. URL https://doi.org/10.1080/03610928108828173. [p713]

M. L. Tiku and R. P. Suresh. A new method of estimation for location and scale parameters. *Journal of Statistical Planning and Inference*, 30(2):281–292, 1992. URL https://doi.org/10.1016/0378-3758(92)90088-A. [p715]

M. L. Tiku and B. Sürücü. Mmles are as good as m-estimators or better. *Statistics & probability letters*, 79(7):984–989, 2009. URL https://doi.org/10.1016/j.spl.2008.12.001. [p714, 716]

D. C. Vaughan. On the tiku-suresh method of estimation. *Communications in Statistics-Theory and Methods*, 21(2):451–469, 1992. URL https://doi.org/10.1080/03610929208830788. [p714]

Y. Y. Wang. Probabilities of the type i errors of the welch tests for the behrens-fisher problem. *Journal of the American Statistical Association*, 66(335):605–608, 1971. URL https://doi.org/10.2307/2283538. [p713]

B. L. Welch. The significance of the difference between two means when the population variances are unequal. *Biometrika*, 29:350–362, 1938. URL https://doi.org/10.2307/2332010. [p713, 716]

K. K. Yuen. The two-sample trimmed t for unequal population variances. *Biometrika*, 61(1):165–170, 1974. URL https://doi.org/10.1093/biomet/61.1.165. [p713]

Gamze Güven
Department of Statistics
Eskisehir Osmangazi University
Eskisehir, Turkey
ORCiD: 0000-0002-8821-3179
gamzeguven@ogu.edu.tr

Şükrü Acıtaş
Department of Statistics
Eskisehir Technical University
Eskisehir, Turkey
ORCiD: 0000-0002-4131-0086
sacitas@eskisehir.edu.tr

Hatice Şamkar
Department of Statistics
Eskisehir Osmangazi University
Eskisehir, Turkey
ORCiD: 0000-0003-1400-2392
hfidan@ogu.edu.tr

Birdal Şenoğlu
Department of Statistics
Ankara University

Ankara, Turkey
ORCiD: 0000-0003-3707-2393
senoglu@science.ankara.edu.tr

		Model (a)			Model (b)		
n_1	n_2	*RW*	*RF*	*W*	*RW*	*RF*	*W*
6	6	0.023	0.014	0.015	0.031	0.025	0.019
6	10	0.026	0.016	0.017	0.033	0.027	0.020
10	10	0.025	0.020	0.018	0.031	0.027	0.020
10	15	0.023	0.016	0.017	0.029	0.027	0.020
10	30	0.030	0.025	0.020	0.029	0.028	0.020
20	20	0.024	0.022	0.020	0.028	0.026	0.021
20	30	0.028	0.025	0.024	0.027	0.026	0.022
20	50	0.026	0.024	0.021	0.028	0.027	0.020
30	50	0.027	0.025	0.021	0.026	0.026	0.022
50	50	0.030	0.028	0.020	0.027	0.026	0.020
		Model (c)			Model (d)		
n_1	n_2	*RW*	*RF*	*W*	*RW*	*RF*	*W*
6	6	0.035	0.024	0.022	0.047	0.036	0.030
6	10	0.037	0.030	0.020	0.053	0.047	0.037
10	10	0.030	0.025	0.020	0.046	0.042	0.033
10	15	0.031	0.026	0.018	0.046	0.043	0.033
10	30	0.031	0.029	0.022	0.046	0.044	0.030
20	20	0.030	0.027	0.019	0.042	0.040	0.027
20	30	0.030	0.028	0.021	0.046	0.044	0.030
20	50	0.030	0.030	0.022	0.045	0.044	0.030
30	50	0.031	0.029	0.020	0.042	0.041	0.026
50	50	0.030	0.028	0.019	0.044	0.044	0.025
		Model (e)			Model (f)		
n_1	n_2	*RW*	*RF*	*W*	*RW*	*RF*	*W*
6	6	0.040	0.030	0.034	0.050	0.042	0.042
6	10	0.050	0.045	0.035	0.054	0.042	0.046
10	10	0.044	0.038	0.038	0.049	0.043	0.044
10	15	0.045	0.042	0.033	0.054	0.048	0.049
10	30	0.042	0.040	0.037	0.055	0.052	0.051
20	20	0.044	0.042	0.028	0.054	0.049	0.047
20	30	0.040	0.037	0.038	0.052	0.049	0.048
20	50	0.041	0.041	0.026	0.051	0.049	0.046
30	50	0.043	0.042	0.036	0.053	0.053	0.049
50	50	0.044	0.043	0.028	0.054	0.052	0.049
		Model (g)			Model (h)		
n_1	n_2	*RW*	*RF*	*W*	*RW*	*RF*	*W*
6	6	0.053	0.041	0.045	0.048	0.032	0.044
6	10	0.056	0.052	0.051	0.044	0.034	0.043
10	10	0.055	0.048	0.047	0.044	0.036	0.042
10	15	0.055	0.052	0.048	0.045	0.039	0.044
10	30	0.052	0.050	0.044	0.045	0.039	0.045
20	20	0.054	0.053	0.049	0.044	0.041	0.044
20	30	0.054	0.053	0.048	0.047	0.044	0.047
20	50	0.055	0.055	0.049	0.050	0.046	0.049
30	50	0.054	0.054	0.048	0.046	0.045	0.046
50	50	0.054	0.053	0.049	0.054	0.051	0.052

Table 1: Simulated Type I error rates of the *RW*, *RF* and *W* tests for Models (a)-(h).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.023	0.014	0.015		0.00	0.024	0.022	0.020
	1.60	0.19	0.15	0.11		0.60	0.10	0.09	0.04
$n = (6, 6)$	3.20	0.51	0.46	0.30	$n = (20, 20)$	1.20	0.35	0.33	0.09
	4.80	0.74	0.70	0.46		1.80	0.65	0.64	0.17
	6.40	0.84	0.82	0.56		2.40	0.84	0.83	0.25
	8.00	0.91	0.89	0.64		3.00	0.94	0.94	0.33
	d	RW	RF	W		d	RW	RF	W
	0.00	0.026	0.016	0.017		0.00	0.028	0.025	0.024
	1.50	0.21	0.17	0.10		0.50	0.09	0.08	0.03
$n = (6, 10)$	3.00	0.57	0.53	0.29	$n = (20, 30)$	1.00	0.31	0.29	0.07
	4.50	0.78	0.76	0.44		1.50	0.59	0.58	0.13
	6.00	0.89	0.88	0.57		2.00	0.80	0.79	0.20
	7.50	0.94	0.93	0.62		2.50	0.92	0.91	0.27
	d	RW	RF	W		d	RW	RF	W
	0.00	0.025	0.020	0.018		0.00	0.026	0.024	0.021
	1.00	0.13	0.11	0.06		0.46	0.10	0.09	0.04
$n = (10, 10)$	2.00	0.43	0.39	0.18	$n = (20, 50)$	0.92	0.32	0.30	0.06
	3.00	0.70	0.67	0.32		1.38	0.60	0.59	0.12
	4.00	0.85	0.83	0.42		1.84	0.81	0.81	0.18
	5.00	0.92	0.91	0.51		2.30	0.92	0.92	0.25
	d	RW	RF	W		d	RW	RF	W
	0.00	0.023	0.016	0.017		0.00	0.027	0.025	0.021
	0.80	0.11	0.09	0.05		0.40	0.09	0.09	0.03
$n = (10, 15)$	1.60	0.36	0.33	0.14	$n = (30, 50)$	0.80	0.30	0.29	0.06
	2.40	0.64	0.61	0.24		1.20	0.60	0.59	0.10
	3.20	0.80	0.79	0.35		1.60	0.83	0.83	0.16
	4.00	0.90	0.89	0.43		2.00	0.94	0.94	0.22
	d	RW	RF	W		d	RW	RF	W
	0.00	0.030	0.025	0.020		0.00	0.030	0.028	0.020
	0.70	0.12	0.10	0.05		0.32	0.08	0.08	0.030
$n = (10, 30)$	1.40	0.37	0.35	0.11	$n = (50, 50)$	0.64	0.26	0.26	0.05
	2.10	0.65	0.63	0.21		0.96	0.55	0.54	0.08
	2.80	0.80	0.79	0.30		1.28	0.79	0.78	0.11
	3.50	0.90	0.89	0.39		1.60	0.93	0.92	0.16

Table 2: Simulated powers of the RW, RF and W tests for Model (a).

	d	*RW*	*RF*	*W*		*d*	*RW*	*RF*	*W*
	0.00	0.031	0.025	0.019		0.00	0.028	0.026	0.021
	5.40	0.22	0.19	0.13		2.00	0.11	0.11	0.04
$n = (6, 6)$	10.80	0.52	0.50	0.34	$n = (20, 20)$	4.00	0.34	0.34	0.11
	16.20	0.72	0.71	0.49		6.00	0.61	0.60	0.19
	21.60	0.83	0.83	0.60		8.00	0.80	0.79	0.30
	27.00	0.90	0.90	0.68		10.00	0.90	0.90	0.38
	d	*RW*	*RF*	*W*		*d*	*RW*	*RF*	*W*
	0.00	0.033	0.027	0.020		0.00	0.027	0.026	0.022
	5.30	0.22	0.21	0.13		2.00	0.10	0.10	0.05
$n = (6, 10)$	10.60	0.53	0.52	0.35	$n = (20, 30)$	4.00	0.33	0.33	0.10
	15.90	0.72	0.71	0.50		6.00	0.62	0.62	0.20
	21.20	0.82	0.82	0.59		8.00	0.81	0.80	0.29
	26.50	0.90	0.90	0.68		10.00	0.90	0.90	0.37
	d	*RW*	*RF*	*W*		*d*	*RW*	*RF*	*W*
	0.00	0.031	0.027	0.020		0.00	0.028	0.027	0.20
	3.60	0.17	0.16	0.09		2.00	0.11	0.11	0.05
$n = (10, 10)$	7.20	0.47	0.46	0.23	$n = (20, 50)$	4.00	0.35	0.34	0.11
	10.80	0.71	0.71	0.38		6.00	0.62	0.62	0.20
	14.40	0.84	0.84	0.49		8.00	0.80	0.80	0.29
	18.00	0.92	0.92	0.58		10.00	0.91	0.91	0.37
	d	*RW*	*RF*	*W*		*d*	*RW*	*RF*	*W*
	0.00	0.029	0.027	0.020		0.00	0.026	0.026	0.022
	3.60	0.17	0.16	0.09		1.60	0.10	0.10	0.04
$n = (10, 15)$	7.20	0.49	0.48	0.23	$n = (30, 50)$	3.20	0.32	0.32	0.08
	10.80	0.72	0.72	0.39		4.80	0.62	0.62	0.15
	14.40	0.85	0.85	0.50		6.40	0.82	0.82	0.23
	18.00	0.92	0.91	0.57		8.00	0.92	0.92	0.30
	d	*RW*	*RF*	*W*		*d*	*RW*	*RF*	*W*
	0.00	0.029	0.028	0.020		0.00	0.027	0.026	0.020
	3.40	0.16	0.16	0.09		1.12	0.08	0.08	0.03
$n = (10, 30)$	6.80	0.45	0.45	0.22	$n = (50, 50)$	2.24	0.26	0.26	0.05
	10.20	0.69	0.69	0.36		3.36	0.53	0.53	0.09
	13.60	0.83	0.83	0.47		4.48	0.76	0.76	0.13
	17.00	0.90	0.90	0.55		5.60	0.90	0.90	0.18

Table 3: Simulated powers of the *RW*, *RF* and *W* tests for Model (b).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.035	0.024	0.022		0.00	0.030	0.027	0.019
	5.00	0.23	0.20	0.14		1.70	0.10	0.10	0.04
$n = (6, 6)$	10.00	0.55	0.52	0.37	$n = (20, 20)$	3.40	0.33	0.32	0.11
	15.00	0.76	0.75	0.53		5.10	0.60	0.59	0.19
	20.00	0.87	0.86	0.62		6.80	0.80	0.80	0.29
	25.00	0.92	0.92	0.70		8.50	0.92	0.91	0.38
	d	RW	RF	W		d	RW	RF	W
	0.00	0.037	0.030	0.020		0.00	0.030	0.028	0.021
	4.80	0.23	0.21	0.14		1.70	0.10	0.10	0.04
$n = (6, 10)$	9.60	0.56	0.54	0.36	$n = (20, 30)$	3.40	0.33	0.33	0.11
	14.40	0.75	0.74	0.52		5.10	0.62	0.61	0.20
	19.20	0.86	0.86	0.62		6.80	0.81	0.81	0.30
	24.00	0.92	0.92	0.69		8.50	0.92	0.92	0.37
	d	RW	RF	W		d	RW	RF	W
	0.00	0.030	0.025	0.020		0.00	0.030	0.030	0.022
	3.00	0.15	0.14	0.08		1.70	0.11	0.11	0.05
$n = (10, 10)$	6.00	0.44	0.42	0.22	$n = (20, 50)$	3.40	0.35	0.35	0.11
	9.00	0.71	0.69	0.37		5.10	0.63	0.63	0.20
	12.00	0.84	0.84	0.48		6.80	0.82	0.82	0.30
	15.00	0.92	0.92	0.56		8.50	0.92	0.92	0.38
	d	RW	RF	W		d	RW	RF	W
	0.00	0.031	0.026	0.018		0.00	0.031	0.029	0.20
	2.60	0.13	0.12	0.06		1.30	0.09	0.09	0.03
$n = (10, 15)$	5.20	0.38	0.37	0.18	$n = (30, 50)$	2.60	0.31	0.30	0.07
	7.80	0.64	0.63	0.32		3.90	0.58	0.58	0.13
	10.40	0.79	0.79	0.44		5.20	0.79	0.79	0.20
	13.00	0.90	0.90	0.54		6.50	0.92	0.92	0.29
	d	RW	RF	W		d	RW	RF	W
	0.00	0.031	0.029	0.022		0.00	0.030	0.028	0.019
	2.60	0.14	0.13	0.07		0.96	0.08	0.08	0.03
$n = (10, 30)$	5.20	0.39	0.39	0.18	$n = (50, 50)$	1.92	0.25	0.25	0.05
	7.80	0.64	0.64	0.32		2.88	0.54	0.53	0.09
	10.40	0.80	0.80	0.44		3.84	0.78	0.77	0.13
	13.00	0.90	0.90	0.54		4.80	0.91	0.91	0.20

Table 4: Simulated powers of the RW, RF and W tests for Model (c).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.047	0.036	0.030		0.00	0.42	0.040	0.027
	2.00	0.15	0.13	0.11		0.80	0.11	0.10	0.07
$n = (6, 6)$	4.00	0.42	0.40	0.34	$n = (20, 20)$	1.60	0.30	0.29	0.17
	6.00	0.69	0.67	0.57		2.40	0.57	0.56	0.32
	8.00	0.84	0.83	0.71		3.20	0.78	0.78	0.46
	10.00	0.91	0.91	0.78		4.00	0.91	0.91	0.57
	d	RW	RF	W		d	RW	RF	W
	0.00	0.053	0.047	0.037		0.00	0.046	0.044	0.030
	2.00	0.16	0.15	0.12		0.80	0.13	0.13	0.09
$n = (6, 10)$	4.00	0.43	0.42	0.34	$n = (20, 30)$	1.60	0.37	0.36	0.25
	6.00	0.70	0.69	0.59		2.40	0.65	0.65	0.47
	8.00	0.84	0.84	0.72		3.20	0.82	0.82	0.62
	10.00	0.91	0.91	0.79		4.00	0.91	0.91	0.71
	d	RW	RF	W		d	RW	RF	W
	0.00	0.046	0.042	0.033		0.00	0.045	0.044	0.030
	1.30	0.12	0.12	0.09		0.80	0.11	0.11	0.06
$n = (10, 10)$	2.60	0.36	0.34	0.26	$n = (20, 50)$	1.60	0.31	0.31	0.17
	3.90	0.63	0.62	0.46		2.40	0.57	0.56	0.32
	5.20	0.81	0.81	0.60		3.20	0.79	0.79	0.46
	6.50	0.92	0.92	0.71		4.00	0.92	0.92	0.58
	d	RW	RF	W		d	RW	RF	W
	0.00	0.046	0.043	0.033		0.00	0.042	0.041	0.026
	1.30	0.13	0.12	0.09		0.64	0.11	0.11	0.05
$n = (10, 15)$	2.60	0.37	0.36	0.26	$n = (30, 50)$	1.28	0.30	0.30	0.14
	3.90	0.65	0.64	0.47		1.92	0.56	0.56	0.27
	5.20	0.83	0.82	0.62		2.56	0.79	0.79	0.39
	6.50	0.92	0.91	0.71		3.20	0.92	0.92	0.51
	d	RW	RF	W		d	RW	RF	W
	0.00	0.046	0.044	0.030		0.00	0.044	0.044	0.025
	1.30	0.13	0.13	0.09		0.48	0.10	0.10	0.05
$n = (10, 30)$	2.60	0.37	0.36	0.25	$n = (50, 50)$	0.96	0.27	0.27	0.10
	3.90	0.65	0.65	0.47		1.44	0.53	0.53	0.19
	5.20	0.82	0.82	0.62		1.92	0.78	0.77	0.30
	6.50	0.91	0.91	0.71		2.40	0.92	0.92	0.39

Table 5: Simulated powers of the RW, RF and W tests for Model (d).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.040	0.030	0.034		0.00	0.044	0.042	0.028
	2.00	0.16	0.13	0.13		0.80	0.11	0.11	0.07
$n = (6, 6)$	4.00	0.44	0.40	0.38	$n = (20, 20)$	1.60	0.30	0.29	0.17
	6.00	0.70	0.67	0.62		2.40	0.55	0.55	0.31
	8.00	0.84	0.83	0.76		3.20	0.78	0.78	0.46
	10.00	0.92	0.91	0.85		4.00	0.91	0.91	0.57
	d	RW	RF	W		d	RW	RF	W
	0.00	0.050	0.045	0.035		0.00	0.040	0.037	0.038
	1.90	0.15	0.14	0.11		0.80	0.10	0.10	0.08
$n = (6, 10)$	3.80	0.41	0.40	0.33	$n = (20, 30)$	1.60	0.31	0.30	0.23
	5.70	0.67	0.66	0.55		2.40	0.59	0.58	0.42
	7.60	0.83	0.83	0.71		3.20	0.81	0.81	0.60
	9.50	0.90	0.90	0.77		4.00	0.92	0.92	0.74
	d	RW	RF	W		d	RW	RF	W
	0.00	0.044	0.038	0.038		0.00	0.041	0.041	0.026
	1.26	0.12	0.10	0.10		0.80	0.11	0.11	0.06
$n = (10, 10)$	2.52	0.35	0.33	0.29	$n = (20, 50)$	1.60	0.30	0.30	0.17
	3.78	0.62	0.60	0.50		2.40	0.58	0.58	0.32
	5.04	0.80	0.80	0.67		3.20	0.79	0.79	0.46
	6.30	0.91	0.91	0.78		4.00	0.91	0.91	0.57
	d	RW	RF	W		d	RW	RF	W
	0.00	0.045	0.042	0.033		0.00	0.043	0.042	0.041
	1.24	0.12	0.12	0.09		0.64	0.11	0.11	0.08
$n = (10, 15)$	2.48	0.33	0.33	0.23	$n = (30, 50)$	1.28	0.31	0.31	0.21
	3.72	0.60	0.60	0.44		1.92	0.59	0.58	0.38
	4.96	0.80	0.80	0.59		2.56	0.81	0.81	0.56
	6.20	0.90	0.90	0.68		3.20	0.93	0.93	0.70
	d	RW	RF	W		d	RW	RF	W
	0.00	0.042	0.040	0.037		0.00	0.044	0.043	0.028
	1.24	0.13	0.13	0.11		0.48	0.10	0.10	0.05
$n = (10, 30)$	2.48	0.36	0.36	0.29	$n = (50, 50)$	0.96	0.27	0.27	0.10
	3.72	0.63	0.63	0.52		1.44	0.55	0.55	0.20
	4.96	0.81	0.81	0.69		1.92	0.78	0.78	0.30
	6.20	0.91	0.91	0.80		2.40	0.92	0.92	0.41

Table 6: Simulated powers of the RW, RF and W tests for Model (e).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.050	0.042	0.042		0.00	0.054	0.049	0.047
	0.90	0.13	0.10	0.11		0.40	0.11	0.11	0.10
$n = (6, 6)$	1.80	0.33	0.29	0.30	$n = (20, 20)$	0.80	0.27	0.26	0.24
	2.70	0.60	0.54	0.56		1.20	0.53	0.51	0.47
	3.60	0.80	0.77	0.77		1.60	0.76	0.75	0.69
	4.50	0.92	0.90	0.89		2.00	0.90	0.90	0.85
	d	RW	RF	W		d	RW	RF	W
	0.00	0.054	0.042	0.046		0.00	0.052	0.049	0.048
	0.90	0.13	0.11	0.12		0.40	0.11	0.11	0.09
$n = (6, 10)$	1.80	0.37	0.34	0.34	$n = (20, 30)$	0.80	0.29	0.29	0.25
	2.70	0.63	0.59	0.59		1.20	0.55	0.54	0.48
	3.60	0.83	0.81	0.80		1.60	0.77	0.77	0.71
	4.50	0.92	0.92	0.90		2.00	0.92	0.92	0.87
	d	RW	RF	W		d	RW	RF	W
	0.00	0.049	0.043	0.044		0.00	0.051	0.049	0.046
	0.64	0.11	0.10	0.10		0.40	0.12	0.11	0.10
$n = (10, 10)$	1.28	0.32	0.29	0.29	$n = (20, 50)$	0.80	0.30	0.30	0.27
	1.92	0.59	0.55	0.54		1.20	0.57	0.57	0.51
	2.56	0.80	0.78	0.76		1.60	0.79	0.78	0.72
	3.20	0.93	0.92	0.90		2.00	0.92	0.92	0.88
	d	RW	RF	W		d	RW	RF	W
	0.00	0.054	0.048	0.049		0.00	0.053	0.053	0.049
	0.60	0.12	0.11	0.10		0.32	0.11	0.11	0.10
$n = (10, 15)$	1.20	0.31	0.29	0.28	$n = (30, 50)$	0.64	0.29	0.28	0.25
	1.80	0.56	0.54	0.52		0.96	0.55	0.55	0.49
	2.40	0.78	0.77	0.73		1.28	0.78	0.78	0.71
	3.00	0.92	0.91	0.88		1.60	0.92	0.92	0.87
	d	RW	RF	W		d	RW	RF	W
	0.00	0.055	0.052	0.051		0.00	0.054	0.052	0.049
	0.60	0.12	0.11	0.11		0.26	0.11	0.11	0.10
$n = (10, 30)$	1.20	0.32	0.31	0.29	$n = (50, 50)$	0.52	0.30	0.29	0.25
	1.80	0.58	0.57	0.54		0.78	0.55	0.54	0.47
	2.40	0.80	0.79	0.75		1.04	0.79	0.79	0.71
	3.00	0.93	0.92	0.89		1.30	0.93	0.93	0.88

Table 7: Simulated powers of the *RW*, *RF* and *W* tests for Model (f).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.053	0.041	0.045		0.00	0.054	0.053	0.049
	1.90	0.13	0.11	0.12		0.86	0.11	0.11	0.10
$n = (6, 6)$	3.80	0.33	0.29	0.30	$n = (20, 20)$	1.72	0.30	0.29	0.26
	5.70	0.60	0.56	0.57		2.58	0.55	0.54	0.50
	7.60	0.81	0.78	0.78		3.44	0.78	0.78	0.74
	9.50	0.92	0.91	0.90		4.30	0.92	0.91	0.89
	d	RW	RF	W		d	RW	RF	W
	0.00	0.056	0.052	0.051		0.00	0.054	0.053	0.048
	1.90	0.13	0.12	0.11		0.86	0.12	0.11	0.10
$n = (6, 10)$	3.80	0.35	0.33	0.32	$n = (20, 30)$	1.72	0.30	0.30	0.27
	5.70	0.61	0.59	0.58		2.58	0.56	0.56	0.51
	7.60	0.82	0.81	0.80		3.44	0.80	0.79	0.76
	9.50	0.93	0.92	0.91		4.30	0.92	0.92	0.90
	d	RW	RF	W		d	RW	RF	W
	0.00	0.055	0.048	0.047		0.00	0.055	0.055	0.049
	1.24	0.11	0.10	0.10		0.82	0.11	0.11	0.10
$n = (10, 10)$	2.48	0.28	0.27	0.25	$n = (20, 50)$	1.64	0.29	0.29	0.26
	3.72	0.53	0.51	0.49		2.46	0.53	0.53	0.49
	4.96	0.75	0.74	0.72		3.28	0.76	0.75	0.71
	6.20	0.90	0.89	0.87		4.10	0.90	0.90	0.87
	d	RW	RF	W		d	RW	RF	W
	0.00	0.055	0.052	0.048		0.00	0.054	0.054	0.048
	1.24	0.11	0.11	0.10		0.64	0.11	0.11	0.10
$n = (10, 15)$	2.48	0.29	0.28	0.27	$n = (30, 50)$	1.28	0.27	0.27	0.24
	3.72	0.54	0.53	0.50		1.92	0.50	0.50	0.46
	4.96	0.76	0.75	0.72		2.56	0.74	0.74	0.70
	6.20	0.90	0.90	0.88		3.20	0.89	0.89	0.86
	d	RW	RF	W		d	RW	RF	W
	0.00	0.052	0.050	0.044		0.00	0.054	0.053	0.049
	1.24	0.12	0.11	0.10		0.50	0.11	0.11	0.09
$n = (10, 30)$	2.48	0.30	0.29	0.27	$n = (50, 50)$	1.00	0.26	0.26	0.23
	3.72	0.55	0.54	0.51		1.50	0.50	0.49	0.45
	4.96	0.76	0.76	0.73		2.00	0.73	0.73	0.68
	6.20	0.90	0.90	0.88		2.50	0.90	0.89	0.86

Table 8: Simulated powers of the *RW*, *RF* and *W* tests for Model (g).

	d	RW	RF	W		d	RW	RF	W
	0.00	0.048	0.032	0.044		0.00	0.044	0.041	0.044
	1.16	0.13	0.10	0.12		0.54	0.11	0.10	0.10
$n = (6, 6)$	2.32	0.35	0.30	0.31	$n = (20, 20)$	1.08	0.30	0.29	0.26
	3.48	0.61	0.56	0.56		1.62	0.57	0.56	0.49
	4.64	0.80	0.77	0.76		2.16	0.79	0.79	0.71
	5.80	0.91	0.90	0.88		2.70	0.93	0.92	0.86
	d	RW	RF	W		d	RW	RF	W
	0.00	0.044	0.034	0.043		0.00	0.047	0.044	0.047
	0.84	0.11	0.09	0.10		0.44	0.11	0.10	0.09
$n = (6, 10)$	1.68	0.30	0.26	0.27	$n = (20, 30)$	0.88	0.28	0.27	0.24
	2.52	0.57	0.52	0.51		1.32	0.56	0.54	0.47
	3.36	0.78	0.74	0.72		1.76	0.79	0.78	0.69
	4.20	0.91	0.89	0.69		2.20	0.92	0.92	0.86
	d	RW	RF	W		d	RW	RF	W
	0.00	0.044	0.036	0.042		0.00	0.050	0.046	0.049
	0.80	0.12	0.10	0.11		0.34	0.10	0.09	0.09
$n = (10, 10)$	1.60	0.32	0.30	0.28	$n = (20, 50)$	0.68	0.27	0.26	0.23
	2.40	0.57	0.55	0.50		1.02	0.50	0.49	0.42
	3.20	0.79	0.77	0.73		1.36	0.74	0.73	0.64
	4.00	0.91	0.90	0.86		1.70	0.90	0.89	0.82
	d	RW	RF	W		d	RW	RF	W
	0.00	0.045	0.039	0.044		0.00	0.046	0.045	0.046
	0.64	0.11	0.10	0.10		0.34	0.11	0.10	0.09
$n = (10, 15)$	1.28	0.29	0.27	0.25	$n = (30, 50)$	0.68	0.29	0.28	0.24
	1.92	0.55	0.52	0.48		1.02	0.55	0.54	0.45
	2.56	0.78	0.76	0.71		1.36	0.79	0.78	0.69
	3.20	0.91	0.90	0.85		1.70	0.92	0.92	0.85
	d	RW	RF	W		d	RW	RF	W
	0.00	0.045	0.039	0.045		0.00	0.054	0.051	0.052
	0.48	0.11	0.09	0.10		0.30	0.10	0.09	0.08
$n = (10, 30)$	0.96	0.28	0.26	0.24	$n = (50, 50)$	0.60	0.25	0.25	0.21
	1.44	0.54	0.51	0.46		0.90	0.48	0.48	0.40
	1.92	0.77	0.75	0.68		1.20	0.73	0.72	0.62
	2.40	0.91	0.90	0.85		1.50	0.90	0.89	0.80

Table 9: Simulated powers of the *RW*, *RF* and *W* tests for Model (h).

Changes in R

by Tomas Kalibera, Sebastian Meyer, Kurt Hornik, Gennadiy Starostin and Luke Tierney

Abstract We present important changes in the development version of R (referred to as R-devel, to become R 4.2) and give a summary of the new search engine interfaced by `RSiteSearch()`. Some statistics on bug tracking activities in 2021 are also provided.

R-devel selected changes

R 4.2.0 is due to be released around April 2022. The following gives a selection of the most important changes in R-devel, which are likely to appear in the new release.

Native UTF-8 support and other changes on Windows

R on Windows now uses UTF-8 as the native encoding. This feature requires recent Windows 10 or newer (or Windows Server 2022 or newer). On older systems, a (non-Unicode) system locale encoding will be used as in earlier versions of R. With this feature, it is now possible to work with characters not representable in the locale encoding (e.g., with Asian characters on European locales). Previously, such characters could only be used with considerable care needed to prevent their mis-representation or undesirable substitution. It is now possible to use Unicode characters even in Rterm, the console front-end for R.

To make this possible, R switched to the Universal C Runtime (UCRT), which is the new C library on Windows and has to be installed manually on Windows 8.1 and older. The switch required a new toolchain targeting UCRT. All code linked statically to R or R packages has to be rebuilt. Therefore, a new toolchain bundle, Rtools42, has been created which includes a recent GCC 10 compiler toolchain targeting 64-bit UCRT and a set of pre-compiled static libraries for R packages. R and CRAN use this new toolchain for R-devel (to become R 4.2.0). Older versions of R will still use older toolchains. As from 4.2, R on Windows will no longer support 32-bit builds. Rtools42, containing only the 64-bit toolchain, is one step simpler to install for users than the earlier toolchain bundle.

The change so far required updates of over 100 CRAN packages and several of their Bioconductor dependencies. As these packages have a very large number of reverse dependencies (packages depending recursively on them), R gained support for automated installation-time patching of packages, so that packages can be quickly patched and their reverse dependencies tested, giving package authors more time to incorporate the updates. This feature is experimental and may be removed in the future.

R allows package authors to maintain the same package sources for R 4.2 (Rtools42) and R 4.1 (Rtools40) by supporting ‘Makevars.ucrt’ and other make/configuration files with extension ‘.ucrt’ which are used by R 4.2 in preference of their existing ‘.win’ variants, but ignored by older versions of R. Both toolchain bundles can coexist on the same machine.

The work on the toolchain and on testing CRAN packages has lead to the discovery of new bugs in GCC: invalid unwind tables causing crashes (GCC PR#103274), inconsistency in option handling related to unwind tables (GCC PR#103465) and lack of support for UCRT/C99 format strings (GCC PR#95130). Additional bugs were found that turned out to be fixed already in later versions of GCC, but required a back-port (GCC PR#101238, GCC PR#100402). Thanks to MinGW-W64 developer Martin Storsjo and GCC developers Eric Botcazou and Martin Liska for their help with identifying and resolving the issues. The Rtools42 toolchain bundle includes patches for these and other, smaller, issues.

Following the philosophy that disruptive changes for users and package authors should be rare, this seemed a good time to change also the default personal library location. Now it is a subdirectory of the Local Application Data directory (usually a hidden directory `C:\Users\username\AppData\Local`). This is to follow Windows conventions, but also to avoid problems users experienced with various cloud backup/syncing services enabled by default for the personal directory (usually `C:\Users\username\Documents`). For the very same reason, the default installation location for user-only installation has been changed to `C:\Users\username\AppData\Local\Programs`.

Additional bug fixes (e.g., for handling previously untested code paths involving characters not representable in system locale encoding) and improvements (e.g., removal of workarounds no longer needed with UCRT) are being added following testing and reports from package authors and are to appear in R 4.2.

More details on the changes in R for Windows and on what is required from package authors are available in Tomas Kalibera et al. blog post and material linked from there.

Graphics changes

Support for isolated groups, compositing operators, affine transformations, and stroking and filling paths has been added to the R graphics engine. The existing support for masks has also been expanded to include luminance masks. An R-level interface for these new features has been added to the **grid** graphics package. See Paul Murrell's blog post for more details. The changes to the R graphics engine mean that packages that provide graphics devices, such as the **ragg** package, will need to be reinstalled.

Hash tables

Hash tables are data structures used to efficiently map *keys* to *values*. Keys can be simple, such as strings or symbols, or more complex objects, such as environments. Hash tables can be thought of as generalizations of environments that allow more general key objects, though without the notion of a parent table. Like environments, and unlike most objects in R, hash tables are mutable.

Hash tables have been used internally in R for many years, in particular in `match()`, `unique()`, and `duplicate()`, to improve the efficiency of these functions. R-devel now provides an R level interface to the hash table infrastructure used in these functions. The R level interface is provided in package **utils**. New hash tables are created by `hashtab()`; entries are created or modified by `sethash()`, and values are retrieved with `gethash()`. More details are available in the help page for `hashtab()`. The R level interface is based loosely on hash table support in Common Lisp.

A C level interface will eventually be made available in the C API as well. The details are still under development. Comparison of keys typically is based on `identical()`, but can also be based on the memory addresses of keys. Address-equality based tables are most likely to be useful at the C level. For address-based hash tables it may be useful to provide a weak version in which keys are not protected from garbage collection and entries are scheduled for removal once keys are determined to no longer be reachable.

Other selected changes

- `matrix(x,n,m)` now warns in more cases where `length(x)` differs from `n * m`, as suggested by Abby Spurdle and Wolfgang Huber in February 2021 on the R-devel mailing list. This warning can be turned into an error by setting environment variable `_R_CHECK_MATRIX_DATA_` to '`TRUE`': `R CMD check --as-cran` does so unless it is already set.
- `simplify2array()` gains an `except` argument for controlling the exceptions used by `sapply()`.
- R on Windows now uses the system memory allocator. Doug Lea's allocator was used since R 1.2.0 to mitigate performance limitations seen with system allocators on earlier versions of Windows.
- R gains more classed errors. Attempting to subset an object that is not subsettable now signals an error of class `notSubsettableError`, with the non-subsettable object contained in the `object` field of the error condition. Also, subscript-out-of-bounds and stack-overflow errors are now signaled as errors of class, respectively, `subscriptOutOfBoundsError` and `stackOverflowError`.
- New partly experimental `Sys.setLanguage()` utility, solving the main problem of PR #18055.
- Deparsing no longer remaps attribute names `dim`, `dimnames`, `levels`, `names` and `tsp` to historical S-compatible names (which `structure()` maps back).

Bug statistics for 2021

Summaries of bug-related activities over the past year were derived from the database underlying R's Bugzilla system. Overall, 244 new bugs or requests for enhancements were reported, 220 reports were closed, and 1065 comments (on any report) were added by a total of 115 contributors. This amounts to averages of about two new reports and two closures over three days, and three comments per day. All totals are about 30% lower than in 2020, especially the number of closures. High bug activity in 2020 had largely been driven by dedicated efforts of several contributors in reviewing old reports.

Figure 1 shows statistics for the numbers of new reports, closures and comments by calendar month and weekday, respectively, in 2021. The frequency of new reports was relatively stable over the year except for a low in March/April. There tended to be more new reports than closures, but this was reversed in November/December in a revived effort to address old reports. The top 5 components reporters have chosen for their reports were "Low-level", "Language", "Documentation", "Misc", and "Wishlist", which is the same set as in 2020. Many reports are suggestions for enhancements and

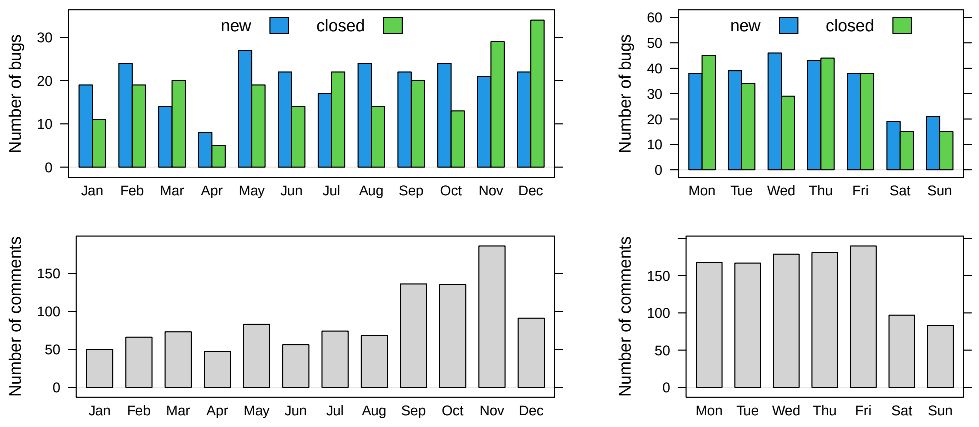


Figure 1: Bug tracking activity by month (left) and weekday (right) in 2021.

marked as Wishlist but are sometimes also put in a specific component, ideally with severity level "enhancement".

Bug discussions led to an average of 65 comments each month from January to August 2021, which is less than in the same period of 2020 with an average of 140 comments each month. Comment activity has increased again in late 2021.

Last but not least, from the numbers by weekday in the right panels of Figure 1 we see that the R community is also active during weekends, though at a lower frequency.

Relaunch of search.R-project.org

A long time ago, Jonathan Baron (University of Pennsylvania, USA) created an "R Site Search" database and has for many years provided a web service for queries into this database, allowing the community to search help files of CRAN packages, task views, vignettes, and initially also the R-help mail archive. This web service was made available as `https://search.R-project.org`, with simple and advanced R interfaces provided by, respectively, functions `RSiteSearch()` in package **utils** and CRAN package **sos** (see the corresponding article on "Searching Help Pages of R Packages" in the R Journal).

The next generation of this web service was developed by Gennadiy Starostin and is now hosted at Wirtschaftsuniversität Wien, Austria. In doing so, there were two major changes.

First, the old service was based on the namazu search engine (`http://www.namazu.org/`), which is no longer actively developed (last release more than ten years old). After careful examination of available open-source alternatives xapian-omega (`https://xapian.org/`) was chosen as the new search engine, which provides the necessary versatility alongside reasonable complexity. The most notable features of xapian are ranked search, phrase and proximity searching, Boolean search operators, Boolean filters, support for stemming of search terms, and allowing simultaneous update and searching.

For compatibility reasons the server still supports requests in the previously used namazu format (limited to the parameters used by the former search engine). This compatibility feature may be dropped in the future.

In addition to the human-readable output of search results, two other formats are made available: "xml" and "opensearch". Simply change in the URL '`FMT=query`' to either '`FMT=xml`' or '`FMT=opensearch`' when sending a HTTP GET request to the server. One can tailor search queries using additional parameters, see the query part of the URL in the default form and the xapian-omega documentation.

Second, the covered CRAN content was expanded. Currently, there are eight categories, any combination of which can be searched simultaneously:

- R manuals (currently based on the R-patched development branch)
- Help pages of base packages (also from R-patched)
- CRAN packages (5 categories): general info, news, readme files, vignettes, and help pages
- CRAN task views

Although content of the majority of these categories is available on CRAN to read and explore, two of them, the help pages of base and CRAN packages, are additionally generated for `search.R-project.`

org. As of December 22, 2021, in terms of searchable documents they constitute approximately 400,000 out of 450,000 total (about 89%).

Generating this content was not straightforward. HTML content is preferable to PDF content for browsing search results, but the new R help system works best for dynamic HTML (see the corresponding article in the R Journal), whereas for the search service, using static HTML is more appropriate. The code for generating static HTML needed a bit of tweaking by Deepayan Sarkar and Kurt Hornik, and now can (again) be used to provide help files which are good for both searching and browsing.

In the future, search.R-project.org may be expanded with relevant sources outside of CRAN, e.g., the Bioconductor project. Depending on user feedback, which is always welcome, one can expect other improvements.

Acknowledgements

Tomas Kalibera's work on the article and R development has received funding from the Czech Ministry of Education, Youth and Sports from the Czech Operational Programme Research, Development, and Education, under grant agreement No.CZ.02.1.01/0.0/0.0/15_003/0000421, from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, under grant agreement No. 695412, and from the National Science Foundation award 1925644.

Tomas Kalibera
Czech Technical University, Czech Republic
Tomas.Kalibera@R-project.org

Sebastian Meyer
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Sebastian.Meyer@R-project.org

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Gennadiy Starostin
WU Wirtschaftsuniversität Wien, Austria
gennadiy.starostin@wu.ac.at

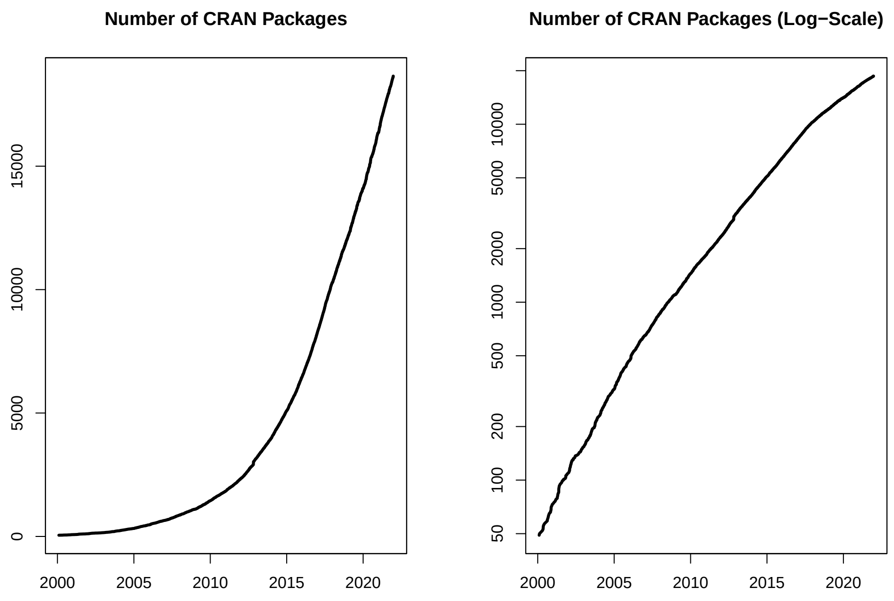
Luke Tierney
University of Iowa, USA
Luke.Tierney@R-project.org

Changes on CRAN

2021-07-01 to 2021-12-31

by Kurt Hornik, Uwe Ligges and Achim Zeileis

In the past 6 months, 1077 new packages were added to the CRAN package repository. 113 packages were unarchived and 331 were archived. The following shows the growth of the number of active packages in the CRAN package repository:



On 2021-12-31, the number of active packages was around 18650.

Changes in the CRAN Repository Policy

The Policy now says the following:

- You can check that the submission was received by looking at `https://CRAN.R-project.org/incoming/`.
- A package showing issues for `macos-arm64` or an 'M1mac' additional issue should be checked using the macbuilder service prior to re-submission.

CRAN package submissions

During the last half of 2021 (July 2021 to December 2021), CRAN received 12256 package submissions. For these, 21622 actions took place of which 14232 (66%) were auto processed actions and 7390 (34%) manual actions.

Minus some special cases, a summary of the auto-processed and manually triggered actions follows:

	archive	inspect	newbies	pending	pretest	publish	recheck	waiting
auto	2748	2722	2577	0	0	3899	1362	924
manual	2760	102	336	464	91	2797	671	169

These include the final decisions for the submissions which were

action	archive	publish
auto	2586 (21.5%)	3336 (27.8%)
manual	2728 (22.7%)	3351 (27.9%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

Interestingly, for the first time in CRAN's history there was a decrease in the number of submissions:

Year	1st half	2nd half
2018	NA	10259
2019	13218	12938
2020	17598	13510
2021	16339	12256

CRAN mirror security

Currently, there are 101 official CRAN mirrors, 83 of which provide both secure downloads via '`https`' *and* use secure mirroring from the CRAN master (via rsync through ssh tunnels). Since the R 3.4.0 release, `chooseCRANmirror()` offers these mirrors in preference to the others which are not fully secured (yet).

CRAN Task View Initiative

To facilitate the maintenance of established CRAN task views as well as the proposal of new ones, a new improved and much more transparent workflow has been established. It is overseen by the newly established *CRAN Task View Editors*: Roger Bivand, Dirk Eddelbuettel, Rocío Joo, David Meyer, Heather Turner, Nathalie Vialaneix, and Achim Zeileis. More details can be found in the corresponding organization on GitHub: `https://github.com/cran-task-views/ctv/`. Currently, the focus is on the transition of the established task views to the new workflow which also involves the archival of some task views which turned out to be too broad to be maintainable (*Graphics* and *SocialSciences*). Also, the *gR* task view has been renamed to *GraphicalModels*. When the transition is completed, a more detailed introduction with further details and instructions will be published soon.

New packages in CRAN task views

Bayesian **BayesianTools**, **MHadaptive**, **RoBMA**.

Cluster **factoextra**.

Databases **dittodb**.

DifferentialEquations **diffeqr**.

Econometrics **pdynmc**, **ssmrob**.

Finance **DOSPortfolio**, **HDShOP**, **RTL**, **bidask**, **etrm**, **greeks**, **ichimoku**, **monobin**, **strand**.

FunctionalData **MFPCA**, **registr**.

Hydrology **HBV.IANIGLA**, **NPRED**, **RavenR**, **WASP**, **hydropeak**, **hydrotoolbox**, **metR**, **nhdR**, **nhdplusTools**, **prism**.

MachineLearning **abess***, **islasso**, **joinet**, **mpath**, **torch**.

MetaAnalysis **amanida**, **metadat**, **nmarank**, **ra4bayesmeta**.

MissingData **BMTAR**, **Iscores**, **MGMM**, **cglasso**, **cmfrec**, **mdgc**, **mgm**.

ModelDeployment **lightgbm**.

NumericalMathematics **FixedPoint**, **GramQuad**, **bignum**, **rim**.

OfficialStatistics **SimSurvey**, **eurostat**, **insee**, **rdhs**, **tidyBdE**.

Optimization **gslnls**, **stochQN**.

Psychometrics **DIFplus**, **semtree**.

ReproducibleResearch **Require**, **gt**, **huxtable**, **makepipe**, **pharmaRTF**, **r2rtf**, **reproducible**, **styler**, **unrtf**.

Robust **RobStatTM**.

TimeSeries **BGVAR**, **GlarmaVarSel**, **STFTS**, **brolgar**, **esemifar**, **mrf**, **mvLSW**, **profoc**, **rdbnomics**, **synthesis**, **tsBSS**, **tsdb**, **tssim**, **uGMAR**, **ugatsdb**.

(* = core package)

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Uwe Ligges
TU Dortmund, Germany
Uwe.Ligges@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org

R Foundation News

by Torsten Hothorn

Donations and members

Membership fees and donations received between 2021-07-06 and 2021-12-22.

Donations

Jordan Aharoni (Canada) b-data GmbH (Switzerland) Mark Cachia (Canada) Shalese Fitzgerald (United States) Knut Helge Jensen (Norway) Roger Koenker (United Kingdom) Merck Research Laboratories, Kenilwort (United States) Statistik Aargau, Aarau (Switzerland)

Supporting members

Diogo Almeida (United Arab Emirates) Tim Appelhans (Germany) Christopher Beltz (United States) Gordon Blunt (United Kingdom) Tamara Bozovic (New Zealand) Greg Bukovatz (United States) Gilberto Camara (Brazil) Susan M Carlson (United States) Charles Cowens (United States) Terry Cox (United States) Robin Crockett (United Kingdom) Robert Daly (Australia) Gergely Daroczi (Hungary) Jasja Dekker (Netherlands) Anthony Alan Egerton (Malaysia) Mitch Eppley (United States) cristiano esclapon (Switzerland) Gottfried Fischer (Austria) David Freedman (United States) Keita Fukasawa (Japan) Sven Garbade (Germany) Anne Catherine Gieshoff (Switzerland) Spencer Graves (United States) Jim Gruman (United States) Krushi Gurudu (United States) Bela Hausmann (Austria) Takehiko Hayashi (Japan) Kieran Healy (United States) Adam Hill (United States) Lorenzo Isella (Belgium) Sebastian Jeworutzki (Germany) JUNE KEE KIM (Korea, Republic of) Gen KOBAYASHI (Japan) Miha Kosmac (United Kingdom) Jan Herman Kuiper (United Kingdom) Seungdoe Lee (Korea, Republic of) Mauro Lepore (United States) Thomas Levine (United States) Chin Soon Lim (Singapore) Joseph Luchman (United States) Alexandra Lypynska (United Kingdom) Gilles Marodon (France) Guido Möser (Germany) yoshinobu nakahashi (Japan) Bernard OFFMANN (France) Dan Orsholits (Switzerland) George Ostrouchov (United States) Abdullah Öztop (Turkey) Matt Parker (United States) Elgin Perry (United States) Peter Ruckdeschel (Germany) Dejan Schuster (Germany) Christian Seubert (Austria) Jagat Sheth (United States) Gaurav Singhal (United States) Tobias Strapatsas (Germany) ROBERT Szabo (Sweden) Koray Tascilar (Germany) Nicholas Turner (United States) Philipp Upravitelev (Russian Federation) Mark van der Loo (Netherlands) Marcus Vollmer (Germany) Jaap Walhout (Netherlands) Sandra Ware (Australia) Arne Jonas Warnke (Germany) Vaidotas Zemlys-Balevičius (Lithuania) Lim Zhong Hao (Singapore)

R Foundation Financial Support 2021

The R Foundation financially supported the following projects in 2021: The R Journal, useR! 2020 and 2021, R Developer Guide Project, and CRAN.

Torsten Hothorn
Universität Zürich, Switzerland Torsten.Hothorn@R-project.org

News from the Forwards Taskforce

by Heather Turner

Forwards is an R Foundation taskforce working to widen the participation of under-represented groups in the R project and in related activities, such as the *useR!* conference. This report rounds up activities of the taskforce during the second half of 2022.

Accessibility

Since a number of people have joined the taskforce with a particular interest in accessibility, we have established a new Accessibility Team, led by s gwynn sturdevant and Jonathan Godfrey. The team will address accessibility to R and the R ecosystem in a broad sense, considering barriers faced by people with disabilities, or working in situations with limited internet access, or working in a language other than English, to name some of the major issues.

At their inaugural meeting in November, the team focused on improving documentation and reference materials for screen-reader users and those who prefer reading electronic documents in dark mode. Both issues are best handled by creating documents in HTML format. Di Cook shared a development version of The R Journal's new HTML format, which was welcomed enthusiastically by the group and there was agreement to work together on testing and improving the journal's accessibility.

The work of the new team is a continuation of the work Forwards has done on accessibility in the past, primarily in the context of the useR! conference. For 2021, Forwards members supported the organizers in their efforts to ensure that presentation materials and conference tools were accessible. Further detail is given in the blog posts Making Accessible Presentations at useR! 2021: The Story Behind the Scenes and How to Improve Conference Accessibility for Screen-reader Users - An Interview with Liz Hare. Liz Hare has been working on a joint project with Silvia Canelón as part of the MiR community, promoting good accessibility practices in data visualisation, a project that received funding from Code for Science and Society.

Community engagement

The community team had a reboot in November, with Richard Ngamita stepping up to join Kevin O'Brien as co-lead. The team plans to focus on fostering networks in regions where the R community is less well connected, in particular supporting the work of AfricaR and the new AsiaR community, which has established a Slack Workspace and plans to hold regional online meetups in 2022.

A key form of support is to encourage knowledge-sharing between regions, for example, by nominating speakers from Africa or Asia for speaking opportunities, or by joining events as a guest speaker. In recent months, Forwards members Mine Çetinkaya-Rundel spoke at satRday Nairobi, Heather Turner spoke at the 1st anniversary of R-Ladies Nairobi, and Kevin O'Brien spoke at the 3rd anniversary of Accra R User Ghana.

Several members of the Community team are part of the R-Consortium Diversity & Inclusion Working Group, led by Samantha Toet. This group is currently working on four big projects: a guide to writing a code of conduct; an event organizer checklist; a speaker directory, and a speaker nomination form for R Consortium-affiliated events.

Conferences Team

The conferences team also gained a new lead, Yanina Bellini Saibene, who has been involved in the organization of many R events and conferences. Along with Natalia da Silva and Riva Quiroga, she chaired the LatinR 2021 conference, which took place online in November.

Being online helped them to reach a record 1000 registrants. Highlights of the program included several papers from Spain, a full panel in Portuguese, and a full panel of Latin American women package developers. This is a great advance from the initial R Foundation endorsed event in 2018, which was a satellite to the 47 JAIIO informatics conference, attended by 50 people.

In collaboration with Claudia Alejandra Huaylla from the Surveys team, Yanina Bellini Saibene reflected in a blog post on the Latin American Community at useR! 2021, describing an exceptional increase in participation compared to previous years. They identify many practices that facilitated this growth, starting with including Latin American R users in the organizing team.

Several members of Forwards have been involved in the Google Season of Docs project to develop a knowledgebase and information board to support the organization of useR! conferences. In particular, Andrea Sánchez-Tapia (co-lead of the Surveys team) and Noa Tamir (co-lead of the Conferences team) were hired as writers on the project and they made sure that diversity and inclusion were considered throughout the documentation. The project built on a lot of previous work by Forwards, including the annual surveys conducted at useR! and pieces of documentation written by the Conferences team as part of their work to make useR! more inclusive. Forwards will continue to be involved in maintaining these resources, which have already gained attention from the organizers of other R conferences.

On-ramps Team

The Forwards first-contributions repository walks people through making a simple pull request on GitHub, a useful skill for contributing to R packages and other open source software. Zane Dax updated the README with instructions in Spanish in time for Hacktoberfest 2021, which encourages people to make four quality pull requests to public GitHub and/or GitLab repositories during October.

The R Contribution Working Group held an Ideas Incubator over the summer to generate new ideas to work on during 2021/2022. One issue prioritised for attention was improving communications. This led to further development of the R Contribution Site, which is now hosted at `https://contributor.r-project.org/` and linked from the main R Project website. The group has also set up the @R_Contributors Twitter account, for sharing event announcements and other news. The current focus is to run some outreach events related to the R Development Guide. Saranjeet Kaur Bhogal and Heather Turner are leading this project, which is supported by a grant from Code for Science and Society, as part of the Digital Infrastructure Incubator. The idea is that these events will provide an on-ramp into a larger contributor event to run in parallel with useR! 2022.

Package Development Modules

Mine Çetinkaya-Rundel and Emma Rand ran a "train-the-trainer" event in November/December aimed at R-Ladies leaders and others who wanted to use the Forwards package development teaching materials with their user groups. Attendees are starting to plan workshops based on the materials: a course was run in early January 2022 by R-Ladies NYC (lead by Joyce Robbins, assisted by Erin Grand and Emily Dodwell) and a course is currently underway by R-Ladies Remote (lead by Heather Turner and Rita Giordano).

Changes in Membership

New members

We welcome the following members to the taskforce:

- Conferences team: Yanina Bellini Saibene (co-leader)
- Surveys team: Andrea Sánchez-Tapia (co-leader)

Previous members

The following members have stepped down:

- Community team: Madlene Hamilton, Ileena Mitra
- On-ramps team: Jenny Bryan (co-leader)
- Social media team: Lorna Maria Aine (co-leader), Shakirah Nakalungi (co-leader), Wenfeng Qin
- Surveys team: David Meza
- Teaching team: Yizhe Xu

We thank them for their contribution to the taskforce. We also acknowledge the work of Emily Dodwell, who has stepped down as administrator after serving for several years (remaining a member of the Teaching team) – s gwynn sturdevant has taken on this role.

Heather Turner
University of Warwick
UK
Heather.Turner@R-Project.org

News from the Bioconductor Project

by Bioconductor Core Team

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor 3.14 was released on 27 October, 2021. It is compatible with R 4.1.0 and consists of 2083 software packages, 408 experiment data packages, 904 up-to-date annotation packages, and 29 workflows.

The project has developed, over the last several years, the 'AnnotationHub' and 'ExperimentHub' resources for serving and managing genome-scale annotation data, e.g., from the TCGA, NCBI, and Ensembl. At the time of release there were 60134 records in the AnnotationHub, and 6075 ExperimentHub records. See the WaldronLab shiny app to get an overview of the AnnotationHub.

Book production continues in this release. Books are built regularly from source and therefore fully reproducible; an example is the community-developed Orchestrating Single-Cell Analysis with Bioconductor.

The Bioconductor 3.14 release announcement includes descriptions of 89 new software packages, and updates to NEWS files for many additional packages. Start using Bioconductor by installing the most recent version of R and evaluating the commands

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
```

Install additional packages and dependencies, e.g., **SingleCellExperiment**, with

```
BiocManager::install("SingleCellExperiment")
```

Docker images provide a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments.

Key learning resources include:

- bioconductor.org to install, learn, use, and develop Bioconductor packages.
- A list of available software, linking to pages describing each package.
- A question-and-answer style user support site and developer-oriented mailing list.
- A community slack (sign up) for extended technical discussion.
- The F1000Research Bioconductor channel for peer-reviewed Bioconductor work flows.
- The Bioconductor YouTube channel includes recordings of keynote and talks from recent conferences including Bioc2021 and BiocAsia2021, in addition to video recordings of training courses.
- Our package submission repository for open technical review of new packages.

The 2021 Bioconductor conference was held in a virtual format August 4-6, 2021.

In conjunction with the Mexican Bioinformatics Network and the Nodo Nacional de Bioinformática CCG UNAM, the Comunidad de Desarrolladores de Software en Bioinformática held two week-long online workshops addressing development of workflows with RStudio and shiny and analysis of single-cell RNA-seq experiments, August 9-13, 2021.

BiocAsia 2021 was held November 1-4 2021 as a virtual event. The Biopackathon project has many points of contact with Bioconductor and recurs monthly.

The National Human Genome Research Institute's Analysis and Visualization Laboratory (AnVIL) is developing with contributions from Bioconductor core team members. Extensive background material includes a series of recorded workshops.

The Bioconductor project continues to mature as a community. The Technical and Community Advisory Boards provide guidance to ensure that the project addresses leading-edge biological problems with advanced technical approaches, and adopts practices (such as a project-wide Code of Conduct) that encourages all to participate. We look forward to welcoming you!

Bioconductor Core Team
Department of Data Science
Dana Farber Cancer Institute, Boston, MA
`maintainer@bioconductor.org`