The R Journal

Volume 14/3, September 2022

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

Contents

News and Notes

The R Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors.

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 Catherine Hurley

On behalf of the editorial board, I am pleased to present Volume 14 Issue 3 of the R Journal.

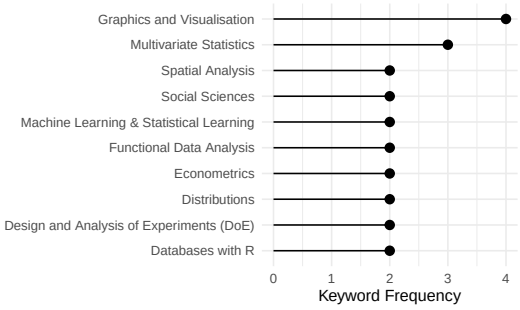
Our incoming editor-in-chief for 2023 Simon Urbanek has been successful in seeking funding from the R Consortium. The project will provide a web-based front-end for managing the R Journal submission and review process.

Behind the scenes, several people assist with the journal operations. Mitchell O'Hara-Wild continues to work on infrastructure, and thanks to this work, producing a new issue is far more straightforward. H. Sherry Zhang continues to develop the **rjtools** package under the direction of Professor Dianne Cook. This package, recently available from CRAN assists in producing RMarkdown articles in the R Journal format. In addition, articles in this issue have been carefully copy edited by Hannah Comiskey.

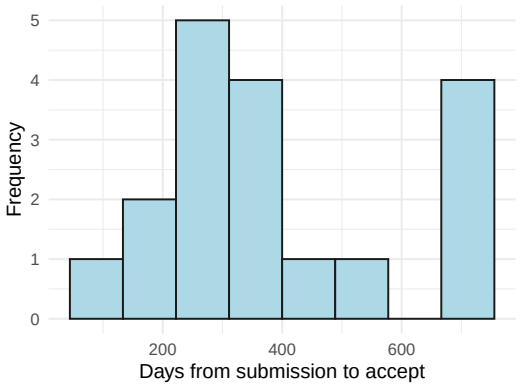
In this issue

News from the CRAN and Bioconductor are included in this issue.

This issue features 18 contributed research articles the majority of which relate to R packages on a diverse range of topics. All packages are available on CRAN. The most common article keywords in this issue are



For the first time, we give times from submission to article acceptabce for an issue. Median times are just under a year, which is consistent other issues over the last few years.



Catherine Hurley
Maynooth University

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

Introducing fastpos: A Fast R Implementation to Find the Critical Point of Stability for a Correlation

by Johannes Titz

Abstract The R package **fastpos** provides a fast algorithm to estimate the required sample size for a Pearson correlation to *stabilize* (Schönbrodt and Perugini 2013). The stability approach is an innovative alternative to other means of sample size planning, such as power analysis. Although the approach is young, it has already attracted much interest in the research community. Still, to date, there exists no easy way to use the stability approach because there is no analytical solution and a simulation approach is computationally expensive with a quadratic time complexity. The presented package overcomes this limitation by speeding up the calculation of correlations and achieving linear time complexity. For typical parameters, the theoretical speedup is around a factor of 250, which was empirically confirmed in a comparison with the original implementation `corEvol`. This speedup allows practitioners to use the stability approach to plan for sample size and theoreticians to further explore the method.

1 Sample size planning with the stability approach

Sample size planning is one of the most crucial steps before conducting an empirical study. The approach-avoidance conflict lies in the desire for reliable conclusions, but the unwillingness to spend resources for large samples. To balance benefit and cost there exist three more or less established paths: power analysis (e.g. Cohen 1988), accuracy in parameter estimation [AIPE; e.g. Maxwell, Kelley, and Rausch (2008)] and interval based accuracy methods (Algina and Olejnik 2003). Recently, a fourth way was introduced: stability (Schönbrodt and Perugini 2013). The general idea of this approach is to determine the sample size at which a certain percentage of studies will fall into an priori specified interval and stay in this interval if the sample size is increased further. For instance, if the population correlation is 0.5, one can define the limits to be 0.4 and 0.6. Given these constraints, what sample size is required to guarantee, with a certain probability (e.g. 90%), that the correlation coefficient will not drop below 0.4 or rise above 0.6 if more participants are added. This sample size is also referred to as the *critical point of stability* for the specific parameters. The stability approach is promising because it (1) focuses on the effect size instead of significance and (2) is fairly intuitive. Indeed, the interest in the method is growing, evident in more than 1500 citations of the original publication. But a proper software package for the stability approach is still missing.

When the concept was introduced, the authors presented a collection of R scripts (`corEvol`, available at a github repository: `https://github.com/nicebread/corEvol`) to derive a sample size table for certain parameters. This implementation is too slow to plan the sample size for an individual study as it can take hours to get reliable results. In this article a faster implementation of the stability approach is introduced available in the R package **fastpos** with the function `find_critical_pos`.

2 Model and implementations

The general model can be shortly described as follows: Define a population correlation ρ, the corridor of stability with lower limit l and upper limit u and a confidence $1 - \alpha$. Now, pairs of values from a bivariate normal distribution with correlation ρ are drawn. In a first step n_{min} pairs are drawn, to which, repeatedly, one more pair is added so that the sample size n is sequentially increased by 1. For every n the correlation r_n is calculated. The point of stability n_{pos} can be described as:

$$n_{\text{pos}} = \min \{n \in \mathbb{N} | l \leq r_m \leq u, \forall m \geq n\} \tag{1}$$

Meaning that the corridor of stability is not left again after the point of stability has been crossed and that the corridor of stability was just entered at the point of stability. Note that n_{pos} is a random variable that has to be evaluated with respect to the normal bivariate distribution. The *critical* point of stability is the quantile $1 - \alpha$ of the probability density function of n_{pos}. It is possible to calculate the transition probabilities of entering, leaving or staying in the corridor of stability for two neighboring sample sizes n and $n + 1$. But, so far, no analytical solution to calculate the critical point of stability has been proposed.

Instead, Schönbrodt and Perugini (2013) set up a Monte Carlo simulation to produce a sample size table for some parameter combinations. In a simulation a maximum sample size n_{max} has to be chosen. Then, for every n from n_{min} to n_{max} the correlation can be calculated. The point of stability for one simulation study can again be described by the above condition. From many of such studies, the critical point of stability can be estimated for the desired confidence.

In the original implementation, the correlations were calculated from scratch for each n, using the function `cor` from `stats`. This is slow as several millions of correlations have to be calculated for a reliable estimate. The correlations at n and $n+1$ only differ by one pair of values, which can be exploited for speed. Take the sum formula for the correlation coefficient at a specific sample size n:

$$r_n = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}} \tag{2}$$

Several sums are calculated, each consisting of adding up n terms. In `corEvol` this is done for every sample size from the minimum to the maximum one. Thus, the total number of added terms for one sum is:

$$\sum_{n=n_{min}}^{n_{max}} n = \sum_{n=1}^{n_{max}} n - \sum_{n=1}^{n_{min}-1} n = \frac{n_{max}\,(n_{max}+1)}{2} - \frac{(n_{min}-1)\,(n_{min}-1+1)}{2} \tag{3}$$

The variable n_{min} can be ignored as it is usually a small value and could even be set to 2. Furthermore, the number of sums in the correlation formula will be the same for every algorithm and is a constant. Dropping constant factors and lower order terms, the time complexity of the described algorithm is $\mathcal{O}(n_{max}^2)$.

In contrast, **fastpos** calculates the correlation for the maximum sample size first. This requires to add n_{max} numbers for one sum. Then it subtracts one value from this sum to find the correlation for the sample size $n_{max}-1$, which happens repeatedly until the minimum sample size is reached (or the corridor is left). In the worst case, the total number of terms for one sum amounts to:

$$n_{max} + n_{max} - n_{min} \tag{4}$$

Again, dropping constant factors and lower order terms, the time complexity of this algorithm is $\mathcal{O}(n_{max})$. The ratio between the two approaches is:

$$\frac{n_{max}\,(n_{max}+1) - (n_{min}-1)\,n_{min}}{4n_{max} - 2n_{min}} \tag{5}$$

For the typically used n_{max} of 1,000 and n_{min} of 20, a speedup of about 250 can be expected. From a theoretical perspective it is also interesting to study the stability approach with larger values of n_{max}, for which the difference becomes even more pronounced.

The theoretical speedup is only an approximation for several reasons. First, one can stop the algorithm when the corridor is left the first time, which is done in **fastpos** but not in `corEvol`. Second, the main function of **fastpos** was written in C++ (via **Rcpp**, Eddelbuettel et al. 2022), which is much faster than normal R. At the same time, the algorithms contain many more steps than just calculating correlations. For instance, setting up the population with a specific ρ takes some time since it usually consists of a million value pairs. The interface functions to setup the simulations also play a role, especially when the algorithm itself is very fast. Thus, it is necessary to study the speed benefit empirically. But before running a benchmark it will be useful to show (1) how to use **fastpos** in general and (2) that it produces the same estimates as `corEvol`.

3 How to use fastpos

For a simple illustration, imagine you plan an empirical study and believe the population correlation is 0.6. You would be happy to find a *stable* correlation between 0.5 and 0.7 with a probability of 80%. What this means is that there is an 80% chance of finding a correlation between 0.5 and 0.7 and by adding more participants this corridor is not left again. In **fastpos** you can run:

```
library(fastpos)
set.seed(20200219)
find_critical_pos(rho = 0.6, precision_absolute = 0.1, confidence_levels = .8,
                  sample_size_min = 20, sample_size_max = 1e3, n_studies = 1e4)

#>   rho_pop pos.80% sample_size_min sample_size_max lower_limit upper_limit
```

```
#> 1     0.6     104              20            1000         0.5         0.7
#>   n_studies n_not_breached precision_absolute precision_relative
#> 1     10000              0                0.1                 NA
```

This loads the package, sets a seed for reproducibility, and runs the simulation with default parameters (except for the ones specifically set). A progress bar is displayed if run in interactive mode. The result is a critical point of stability of 104.

The main function of the package `find_critical_pos` will usually suffice for most use cases. Its parameters are documented in detail in the package. The population correlation (`rho`) and the number of simulation studies (`n_studies`) is self-explanatory. The chosen precision (`precision_absolute`) of 0.1 (i.e. the half-width) will result in the desired corridor between 0.5 and 0.7. There is also a convenience argument to set the precision as a relative value, `precision_relative`, which will override `precision_absolute`. For instance, `precision_relative = 0.1` produces an interval of $\rho \pm \rho \cdot 0.1$. Alternatively, one can also provide the lower and upper limit of the corridor directly via `lower_limit` and `upper_limit`. This is especially useful if the corridor is not symmetric. Notable, most parameters can also take vectors so it is possible to run multiple simulations for different `rho` values (and corresponding other parameters) at once.

The parameter `confidence_levels` defines the quantile corresponding to the critical point of stability. This parameter can be a single value or a vector, but is fixed for all `rho` values. If different confidence levels are of interest, providing them as a vector saves a lot of resources because one simulation can be used to calculate the critical points of stability for all confidence levels.

The parameters `sample_size_min` and `sample_size_max` set the minimum and maximum sample size of one simulation study. As in `corEvol` they default to 20 and 1,000. This means a sample of 20 observation pairs is drawn from the population and step by step one more observation is added until the sample size of 1,000 is reached.

The output summarizes the individually set (and default) parameters as well as the critical point of stability of about 104. The value will change slightly from run to run because only 10,000 simulations are done here. In practice one can make a quick estimate with the default parameters and then increase the number of simulation studies for a more robust estimate. Under GNU/Linux one can also take advantage of the multicore support (parameter `n_cores`). This functionality is currently implemented via the **pbmcapply** package (Kuang, Kong, and Napolitano 2022), which is based on `parallel`.[1]

For another illustration let us reproduce Schönbrodt and Perugini (2013)'s oft-cited table of the critical points of stability for an absolute precision of 0.1 (meaning that the corridor will be $\rho \pm .1$). We take advantage of the vectorized input option by providing several ρ values at once. Furthermore, we increase the number of studies to 100,000 to get accurate estimates. To cache the simulation results we use **simpleCache** (Nagraj and Sheffield 2021):

```
library(simpleCache)
setCacheDir("titz_cache")
simpleCache("sim2", {find_critical_pos(rho = seq(.1, .7, .1), n_studies = 1e5)})
sim2
```

```
#>   rho_pop pos.80% pos.90% pos.95% sample_size_min sample_size_max lower_limit
#> 1     0.1     253     363     478              20            1000         0.0
#> 2     0.2     237     339     448              20            1000         0.1
#> 3     0.3     212     305     404              20            1000         0.2
#> 4     0.4     181     262     343              20            1000         0.3
#> 5     0.5     143     208     277              20            1000         0.4
#> 6     0.6     103     150     200              20            1000         0.5
#> 7     0.7      65      96     129              20            1000         0.6
#>   upper_limit n_studies n_not_breached precision_absolute precision_relative
#> 1         0.2     1e+05            139                0.1                 NA
#> 2         0.3     1e+05            102                0.1                 NA
#> 3         0.4     1e+05             43                0.1                 NA
#> 4         0.5     1e+05             15                0.1                 NA
#> 5         0.6     1e+05              5                0.1                 NA
#> 6         0.7     1e+05              0                0.1                 NA
#> 7         0.8     1e+05              0                0.1                 NA
```

The results are very close to the original publication (Schönbrodt and Perugini 2013). Note that a warning is shown because in some simulations the point of stability was not found. This is not too

[1]The multicore support will not be demonstrated here because it is difficult to create reproducible examples across different operating systems and number of cores.

surprising as one can easily imagine an extreme outlier study that, for instance, starts at a negative correlation with $n = 20$ and does not reach the specified corridor of stability at the maximum sample size of $n = 1,000$. There are different ways to handle these outliers, which will affect the estimate.

4 Handling outliers

When comparing the table from above with the one in Schönbrodt and Perugini (2013), one should notice that **fastpos** usually produces larger estimates. To illustrate this more reliably we need to increase the number of studies, so that random fluctuations are minimized. Here we will run 100 simulations with 1,000,000 studies each.[2]

```
simpleCache("sim3", {find_critical_pos(rho = rep(0.1, 100),
                                       sample_size_max = 1e3, n_studies = 1e6)})
```

A good summary of the data is the mean and the standard error of the distribution. Before calculating these statistics, we select only the points of stability from the result:

```
sim3 <- sim3[, c("pos.80%", "pos.90%", "pos.95%")]
colMeans(sim3)

#>  pos.80%  pos.90%  pos.95%
#> 253.2020 363.2100 477.5905

round(apply(sim3, 2, sd), 3)

#> pos.80% pos.90% pos.95%
#>   0.603   0.729   1.035
```

The average estimates are 253, 363 and 478 (with reasonably small standard errors), while in Schönbrodt and Perugini (2013) they are 252, 362, 470 and in Schönbrodt and Perugini (2018) 252, 360 and 474. Note that in every case **fastpos** gives a slightly larger estimate, which is not just a random fluctuation but related to the warning. In `corEvol`, if the corridor of stability is not reached, the respective study is ignored when calculating the critical point of stability. This leads to a systematic underestimation of the critical point of stability.

To illustrate this, we can use the lower level functions `create_pop` and `simulate_pos` to create a distribution of points of stability. In the following, the first line creates a population with a specific correlation and the second line produces several points of stability by drawing from this population. In contrast to the main function of the package (`find_critical_pos`), the function `simulate_pos` does not calculate quantiles, but only generates points of stability.

```
pop <- create_pop(rho = 0.1, size = 1e6)
simpleCache("sim4", {simulate_pos(x_pop = pop[, 1], y_pop = pop[, 2],
                                  n_studies = 1e6, sample_size_min = 20,
                                  sample_size_max = 1e3, replace = TRUE,
                                  lower_limit = 0, upper_limit = 0.2,
                                  progress = FALSE)})
```

There are two ways to calculate the quantiles of interest:

```
quantile(sim4, c(.8, .9, .95), na.rm = TRUE)

#> 80% 90% 95%
#> 252 361 473

sim4b <- ifelse(is.na(sim4), 1e3, sim4)
quantile(sim4b, c(.8, .9, .95))

#> 80% 90% 95%
#> 253 363 478
```

[2] It is worth noting that (with a single core) this simulation would take several weeks to complete with `corEvol` but only takes about 66 minutes with **fastpos**.

In the first calculation, the studies that did not reach the corridor of stability are ignored (like in `corEvol`), while in the second calculation it is assumed that the point of stability was reached at the maximum sample size. When repeating this simulation, the values will vary slightly but the second method will never produce smaller estimates. That the second method is more accurate can be tested by increasing the maximum sample size (to avoid studies that do not reach the corridor of stability). Here, we will set the maximum sample size to 5,000:

```
simpleCache("sim5", {find_critical_pos(rho = rep(0.1, 100),
                                       sample_size_max = 5e3,
                                       n_studies = 1e6)})
sim5 <- sim5[, c("pos.80%", "pos.90%", "pos.95%")]
colMeans(sim5)

#>  pos.80%  pos.90%  pos.95%
#> 253.3100 363.5100 478.4305

round(apply(sim5, 2, sd), 3)

#> pos.80% pos.90% pos.95%
#>   0.631   0.870   1.266
```

If every study reaches the point of stability, the estimates are 253, 364 and 478. When the maximum sample size is too small (as in the second to last simulation), **fastpos** is indeed closer to these estimates than `corEvol`. While the difference to `corEvol` might seem practically negligible, `corEvol`'s estimates are clearly biased. Furthermore, depending on the parameters, the problem can become more severe. A very narrow corridor will lead to many studies not reaching the corridor, which `corEvol` will not even notice. On the other hand, **fastpos** will throw a warning, which should be taken seriously.

But even **fastpos** might underestimate the critical point of stability if the maximum sample size is too small: All estimates with a maximum sample size of 5,000 are slightly larger than the ones with a maximum sample size of 1,000. With a larger maximum sample size, there are more opportunities to leave the corridor again. At some point the probability of this event is very low because the corridor limits are too far away, but the probability is not 0. Thus, increasing the maximum sample size even further (here to 10,000) should lead to slightly larger estimates:

```
simpleCache("sim6", {find_critical_pos(rho = rep(0.1, 100),
                                       sample_size_max = 1e4,
                                       n_studies = 1e6)})
sim6 <- sim6[, c("pos.80%", "pos.90%", "pos.95%")]
colMeans(sim6)

#>  pos.80%  pos.90%  pos.95%
#> 253.4000 363.7000 478.7005

round(apply(sim6, 2, sd), 3)

#> pos.80% pos.90% pos.95%
#>   0.667   0.937   1.234
```

Indeed, all estimates are slightly larger but after rounding to a whole number only for the confidence of 95% the critical point of stability changes from 478 to 479. Furthermore, the randomness of the simulations permits such fluctuations since the standard errors are about 1. But note that all estimates increase when the maximum sample size changes from 1,000 to 5,000 and then to 10,000, which is a clear hint for a bias. Nonetheless, it appears unlikely that the estimates would increase much, when the maximum sample size grows further. The remaining problem is that the theoretical idea of stability assumes an *infinite* maximum sample size or, at least, that the maximum sample size is equal to the population size. It is therefore of some technical and practical interest to investigate the relationship between the maximum sample size and the critical point of stability in a dedicated simulation study with **fastpos**. Such a study would not be easy to approach with `corEvol` because of the quadratic time complexity. In the next section the speed difference between both packages is demonstrated empirically in a benchmark.

5 Benchmark

corEvol was written as a script for a simulation study and cannot be simply called via a function in a package. Thus, a helper function will be used that sources the script files. To make the benchmark reproducible, the original repository corEvol was forked and a benchmark branch created. With git and a shell installed, the following tries to update the repository in the corEvol folder. If this is unsuccessful (the folder does not exist), the repository is cloned.

```
git -C corEvol pull || git clone --single-branch --branch benchmark \
  https://github.com/johannes-titz/corEvol
```

Alternatively, you can download the required files from the supplementary material of this article.

For corEvol, two files are sourced for the benchmark. The first file generates the simulations and the second calculates the critical point of stability. In corEvol a simulation run takes a lot of time and thus it is not practical to run it too many times. But since the expected speed difference between both implementations is substantial, this should not be a concern. Here, ten repetitions were done with the **microbenchmark** (Mersmann 2021) package. The code was run on a Dell Server r6515 with an AMD EPYC 7302P CPU. Only one core was used to not confound the result with the specific parallel implementation.

```
library(microbenchmark)
corevol <- function() {
  setwd("corEvol")
  source("01-simdata.R")
  source("02-analyse.R")
  setwd("../")
}
fastpos <- function() {
  find_critical_pos(rho = .1, sample_size_max = 1e3, n_studies = 1e4,
                    progress = FALSE)
}
simpleCache("bm", {microbenchmark(corevol = corevol(), fastpos = fastpos(),
                                  times = 10, unit = "s")})
summary(bm)
```

```
#>      expr         min         lq        mean     median          uq         max
#> 1 corevol 350.4551133 352.642384 355.7306834 355.221224 358.2854179 365.8751542
#> 2 fastpos   0.5692708   0.579922   0.6215005   0.596842   0.6066209   0.8496276
#>   neval cld
#> 1    10   b
#> 2    10  a
```

For the chosen parameters, **fastpos** is about 572 times faster than corEvol, for which there are two main reasons: (1) **fastpos** is built around a C++ function via **Rcpp** and (2) this function does not calculate every calculation from scratch, but only calculates the difference between the correlation at sample size n and $n-1$ via the sum formula of the Pearson correlation (see Equation (2)). There are some other factors that might play a role, but they cannot account for the large difference found. For instance, setting up a population takes quite long in corEvol (about 17s), but compared to the 6 minutes required overall, this is only a small fraction. There are other parts of the corEvol code that are fated to be slow, but again, a speedup by a factor of 572 cannot be achieved by improving these parts. The presented benchmark is not comprehensive, but still demonstrates that **fastpos** can be used with no significant waiting time for a typical scenario, while for corEvol this is not the case.

Another benchmark on a local i5-3320 2.6 GHz CPU from 2012 resulted in means of 1.5s for **fastpos** and 603s for corEvol giving a speedup of around 400. Thus, even on older CPUs and single-cored **fastpos** delivers almost instantly for default parameters.

6 Other effect sizes

The focus of **fastpos** is on the Pearson correlation as the effect size. In principle the stability approach can be extended to all sorts of effect sizes or even other statistical parameters. Since the original authors studied the Pearson correlation, it made sense to improve the algorithm for this specific use

case. But mathematical shortcuts as in Equation (2) should also exist for other effect sizes and might be implemented in the future.

A simple alternative for applying the method to other effect sizes is to convert these effects to the Pearson correlation. Such conversions are very common in meta-analyses, where a consistent effect size must be used across all studies to calculate a meaningful average effect. Standard approximate conversion formulas can be found in text books on research methods (Borenstein et al. 2021; Sedlmeier and Renkewitz 2018). Several packages in R also provide these conversions. For instance, **effectsize** (Ben-Shachar, Lüdecke, and Makowski 2020) includes the functions `d_to_r` and `r_to_d`. `d_to_r` is based on the approximation $r = \frac{d}{d^2+4}$, which should only be used for equal group sizes. As an example, consider $d = 0.5$ between two equally sized groups and a corridor with limits of 0.4 and 0.6.

```
r <- effectsize::d_to_r(0.5)
lower_limit <- effectsize::d_to_r(0.4)
upper_limit <- effectsize::d_to_r(0.6)
simpleCache("sim7", {find_critical_pos(rho = r, sample_size_max = 11e3,
                                       n_studies = 1e5,
                                       lower_limit = lower_limit,
                                       upper_limit = upper_limit)})
sim7
```

```
#>     rho_pop pos.80% pos.90% pos.95% sample_size_min sample_size_max lower_limit
#> 1 0.2425356    1119    1606    2108              20           11000   0.1961161
#>   upper_limit n_studies n_not_breached precision_absolute precision_relative
#> 1   0.2873479     1e+05              0                 NA                 NA
```

The corresponding Pearson correlation for $d = 0.5 \pm 0.1$ is about 0.24, with very narrow and slightly asymmetric limits (0.20 to 0.29). The critical point of stability is 2108 for a confidence level of 95%.

7 Summary

In this article, **fastpos**, a package for estimating the critical point of stability was introduced. The package is much faster than the original implementation and can be conveniently used for sample size planning as well as Monte Carlo simulation studies. While the original implementation ignores studies that do not reach the corridor of stability, **fastpos** takes them into account and gives a more conservative and more accurate estimate (i.e. a larger critical point of stability). From a practitioner's perspective, this detail might be negligible for typical parameters and relatively wide corridors. But from a statistical perspective, this detail is of relevance and further simulation studies are required to better understand the stability approach in general. Finally, a comparison to other methods of sample size planning would be of much interest and could influence how empirical scientists plan for sample size in the future. **fastpos** can be a useful tool to achieve these goals.

8 Acknowledgment

I want to thank Matthias Hörr and Thomas Schäfer for insightful discussions about the stability approach. Furthermore, I want to thank an anonymous reviewer for many helpful suggestions on how to improve the article and the package.

References

Algina, James, and Stephen Olejnik. 2003. "Sample Size Tables for Correlation Analysis with Applications in Partial Correlation and Multiple Regression Analysis." *Multivariate Behavioral Research* 38: 309–23. https://doi.org/10.1207/S15327906MBR3803_02.

Ben-Shachar, Mattan S., Daniel Lüdecke, and Dominique Makowski. 2020. "effectsize: Estimation of Effect Size Indices and Standardized Parameters." *Journal of Open Source Software* 5 (56): 2815. https://doi.org/10.21105/joss.02815.

Borenstein, Michael, Larry V Hedges, Julian PT Higgins, and Hannah R Rothstein. 2021. *Introduction to Meta-Analysis*. John Wiley & Sons.

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Eddelbuettel, Dirk, Romain Francois, JJ Allaire, Kevin Ushey, Qiang Kou, Nathan Russell, Inaki Ucar, Douglas Bates, and John Chambers. 2022. *Rcpp: Seamless r and c++ Integration*. https://CRAN.R-project.org/package=Rcpp.

Kuang, Kevin, Quyu Kong, and Francesco Napolitano. 2022. *pbmcapply: Tracking the Progress of Mc*pply with Progress Bar*. https://CRAN.R-project.org/package=pbmcapply.

Maxwell, S. E., K. Kelley, and J. R. Rausch. 2008. "Sample Size Planning for Statistical Power and Accuracy in Parameter Estimation." *Annual Review of Psychology* 59: 537–63. https://doi.org/10.1146/annurev.psych.59.103006.093735.

Mersmann, Olaf. 2021. *microbenchmark: Accurate Timing Functions*. https://CRAN.R-project.org/package=microbenchmark.

Nagraj, VP, and Nathan Sheffield. 2021. *simpleCache: Simply Caching r Objects*. https://CRAN.R-project.org/package=simpleCache.

Schönbrodt, F. D., and M. Perugini. 2013. "At What Sample Size Do Correlations Stabilize?" *Journal of Research in Personality* 47: 609–12. https://doi.org/10.1016/j.jrp.2013.05.009.

———. 2018. "Corrigendum to 'At What Sample Size Do Correlations Stabilize?' [J. Res. Pers. 47 (2013) 609–612]." *Journal of Research in Personality* 74: 194. https://doi.org/10.1016/j.jrp.2018.02.010.

Sedlmeier, Peter, and Frank Renkewitz. 2018. *Forschungsmethoden und Statistik für Psychologen und Sozialwissenschaftler*. 3rd ed. Hallbergmoos, Germany: Pearson Studium.

Johannes Titz
Chemnitz University of Technology
Department of Psychology
Chemnitz, Germany
https://johannestitz.com
ORCiD: 0000-0002-1102-5719
johannes.titz@psychologie.tu-chemnitz.de

WLinfer: Statistical Inference for Weighted Lindley Distribution

by Yu-Hyeong Jang, SungBum Kim, Hyun-Ju Jung, and Hyoung-Moon Kim

Abstract New distributions are still being suggested for better fitting of a distribution to data, as it is one of the most fundamental problems in terms of the parametric approach. One of such is weighted Lindley (WL) distribution (Ghitany et al., 2011). Even though WL distribution has become increasingly popular as a possible alternative to traditional distributions such as gamma and log normal distributions, fitting it to data has rarely been addressed in existing R packages. This is the reason we present the **WLinfer** package that implements overall statistical inference for WL distribution. In particular, **WLinfer** enables one to conduct the goodness of fit test, point estimation, bias correction, interval estimation, and the likelihood ratio test simply with the 'WL' function which is at the core of this package. To assist users who are unfamiliar with WL distribution, we present a brief review followed by an illustrative example with R codes.

1 Introduction

Weighted Lindley (WL) distribution has recently received considerable attention since it provides a more flexible fit to data from various fields than traditional widely-used distributions such as exponential, log normal, and gamma distributions (Ghitany et al., 2011; Mazucheli et al., 2013). The probability density function (pdf) of WL distribution is given by

$$f(x) = \frac{\lambda^{\phi+1}}{(\lambda+\phi)\Gamma(\phi)} x^{\phi-1}(1+x)\exp(-\lambda x), \quad x>0, \lambda>0, \phi>0,$$

which can be interpreted as a mixture of two gamma distributions

$$f(x) = \frac{\lambda}{\lambda+\phi} f_1(x) + \frac{\phi}{\lambda+\phi} f_2(x), \quad x>0, \lambda>0, \phi>0,$$

where

$$f_i(x) = \frac{\lambda^{\phi+i-1}}{\Gamma(\phi+i-1)} x^{\phi+i-2}\exp(-\lambda x), \quad x>0, \lambda>0, \phi>0, \quad i=1,2.$$

Due to its nature as a gamma mixture, however, statistical inference for WL distribution, such as parameter estimation, bias correction, interval estimation, and statistical test, is overall more cumbersome and tedious than those of the aforementioned distributions.

Despite this difficulty, there is no existing R package that implements this comprehensive process for WL distribution. To the best of our knowledge, **mle.tools** (Mazucheli et al., 2017) is the only R package that enables one to obtain maximum likelihood (ML) estimates for WL distribution with asymptotic variance and bias correction. However, they are just limited to ML estimates. An R package **fitdistrplus** (Delignette-Muller and Dutang, 2015) which fits certain univariate distributions to data sets is not applicable to WL distribution, while **LindleyR** package (Mazucheli et al., 2016) is exclusively for the use of pdf, cumulative distribution function (cdf), quantile, and random number generation, not statistical inference itself.

Based on this motivation, we present an R package **WLinfer** by providing various estimation methods in addition to maximum likelihood estimator (MLE), such as method of moment estimator (MME), modified method of moment estimator (MME_m), and closed form MLE-like estimator (MLE_c). For bias correction, **WLinfer** encompasses Firth's method and the bootstrap method which were not considered in **mle.tools**, as well as Cox and Snell's method. Furthermore, **WLinfer** provides the goodness of fit and likelihood ratio tests.

The remainder of this paper is organized as follows. First, we briefly review the theoretical results for each inferential step ranging from the goodness of fit test to the likelihood ratio test, while introducing relevant arguments of the 'WL' function. We then provide an example to illustrate the whole output of 'WL' function and how to use 'WL' function which implements the whole statistical inference at once. Finally, we conclude with summarizing remarks.

The **WLinfer** package is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=WLinfer. R code for the examples demonstrated herein has been provided as supplementary material. The supplementary code has been tested with **WLinfer** version 1.0.0, and results presented herein have been produced with this version.

2 Goodness of fit test

The first step in fitting a distribution to data is to check whether the data can be assumed as generated from the distribution. For this purpose, many goodness of fit tests have been developed. Among them, **WLinfer** includes three popular tests: the Kolmogorov-Smirnov test, the Anderson-Darling test, and the Cramér-von Mises test. Uncertainty that could occur from using estimates instead of true values are not considered here. The argument `dist_test` of '`WL`' function is specified by one of either `"ks"`, `"cvm"`, `"ad"`, or `"all"`. The default is `dist_test="ks"` while `dist_test=all"` returns the results of all three tests.

3 Point estimation

WLinfer package considers four estimators: MLE, MLE$_c$, MME and MME$_m$. To the best of our knowledge, these are the only estimators of which asymptotic distribution has been analytically proven. In this study, we only review the estimation formulas. Please refer to Kim and Jang (2020); Ghitany et al. (2017); Mazucheli et al. (2013) for more specific theoretical results.

- MLE of WL distribution is obtained by

$$\widehat{\lambda}_{MLE} = \frac{-\widehat{\phi}_{MLE}(\overline{X}-1) + \sqrt{\left[\widehat{\phi}_{MLE}(\overline{X}-1)\right]^2 + 4\widehat{\phi}_{MLE}\left(\widehat{\phi}_{MLE}+1\right)\overline{X}}}{2\overline{X}}$$

  where $\widehat{\phi}_{MLE}$ is the solution of the nonlinear equation

$$\psi\left(\widehat{\phi}_{MLE}\right) + \frac{1}{\widehat{\lambda}_{MLE}+\widehat{\phi}_{MLE}} - \log\left(\widehat{\lambda}_{MLE}\right) - \frac{1}{n}\sum_{i=1}^{n}\log\left(x_i\right) = 0,$$

  with $\psi(x) = \dfrac{\mathrm{d}\log\left(\Gamma(x)\right)}{\mathrm{d}x}$.

- MLE$_c$

$$\widehat{\lambda}_{MLE_c} = \frac{d + \sqrt{d^2 + \frac{4(1+a_1)\cdot a_0(a_0-1)}{(\sum X_i/n)-a_1}}}{2(a_1+1)}, \qquad \widehat{\phi}_{MLE_c} = a_1\widehat{\lambda} - a_0,$$

  where $d = a_0 + \frac{(1+a_1)(1-a_0)-1}{\sum X_i/n - a_1}$, $a_1 = \dfrac{\sum_i^n X_i\log(X_i)}{\sum_i^n \log(X_i)}$ and $a_0 = \dfrac{\sum_i^n \frac{X_i\log(X_i)}{1+X_i} + n}{\sum_i^n \log(X_i)}$.

- MME

$$\widehat{\lambda}_{MME} = \frac{-\widehat{\phi}_{MME}(\overline{X}-1) + \sqrt{\left[\widehat{\phi}_{MME}(\overline{X}-1)\right]^2 + 4\overline{X}\widehat{\phi}_{MME}\left(\widehat{\phi}_{MME}+1\right)}}{2\overline{X}},$$

$$\widehat{\phi}_{MME} = \frac{-g\left(\overline{X},S^2\right) + \sqrt{\left[g\left(\overline{X},S^2\right)\right]^2 + 16S^2\left[S^2+(\overline{X}+1)^2\right]\overline{X}^3}}{2S^2\left[S^2+(\overline{X}+1)^2\right]},$$

  where $g\left(\overline{X},S^2\right) = S^4 - \overline{X}\left(\overline{X}^3 + 2\overline{X}^2 + \overline{X} - 4S^2\right)$, $S^2 = \frac{1}{n}\sum_{i=1}^n\left(X_i-\overline{X}\right)^2$.

- MME$_m$

$$\widehat{\lambda}_{MME_m} = \frac{\overline{X^*}(1-\overline{X^*})}{\overline{X}\cdot\overline{X^*} - (1-\overline{X^*})},$$

$$\widehat{\phi}_{MME_m} = \frac{(1-\overline{X^*})^2}{\overline{X}\cdot\overline{X^*} - (1-\overline{X^*})}, \quad \text{where } X^* = \frac{1}{1+X}.$$

Each estimation method can be implemented by choosing '`est_method`' of '`WL`' function, among `"MLE"`, `"MLEc"`, `"MME"`, and `"MMEm"`. The default refers to '`est_method = "MLEc"`'. If point estimation is solely of interest, one can separately obtain point estimates with the following codes:

```
data(fail_fiber)  # fail_fiber is included in WLinfer package.

MLEc_WL(fail_fiber)  # Closed form MLE-like estimator
MME_WL(fail_fiber)  # Method of moment estimator
MMEm_WL(fail_fiber)  # Modified method of moment estimator
MLE_WL(fail_fiber,init =1)  # Initial value for phi is required.
```

We use `fail_fiber` data set (Bader and Priest, 1982) which is included in the **WLinfer** package. This data set consists of 65 observations of the strength measurement of carbon fiber.

4 Bias correction

Unless sample size is sufficient for achieving consistency, bias correction is highly recommended since all the aforementioned estimators are upwardly biased in the case of small samples (Kim and Jang, 2020; Mazucheli et al., 2013; Wang and Wang, 2017). To this end, the **WLinfer** package provides three bias correction methods: Cox and Snell's method (Cox and Snell, 1968; Cordeiro and Klein, 1994) for MLE and MLE_c, Firth's method (Firth, 1993) for MLE, and the bootstrap method (Efron, 1979; Efron and Tibshirani, 1986) for MLE_c. After a series of tedious calculations, correction formulas are given as follows:

- Cox and Snell's method

$$\widehat{\boldsymbol{\theta}}_{CMLE} = \widehat{\boldsymbol{\theta}}_{MLE} - \begin{pmatrix} \delta(\widehat{\lambda}) \\ \delta(\widehat{\phi}) \end{pmatrix},$$

$$\widehat{\boldsymbol{\theta}}_{CMLE_c} = \widehat{\boldsymbol{\theta}}_{MLE_c} - \begin{pmatrix} \delta(\widehat{\lambda}) \\ \delta(\widehat{\phi}) \end{pmatrix},$$

where $\delta(\widehat{\lambda}) = \frac{1}{n} \cdot \frac{N_1\big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_{MLE_cs}}}{D\big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_{MLE_cs}}}$ and $\delta(\widehat{\phi}) = \frac{1}{n} \cdot \frac{N_2\big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_{MLE_cs}}}{D\big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_{MLE_cs}}}$ with

$$D = \left[\left(\psi'(\phi) - \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{\phi+1}{\lambda^2} - \frac{1}{(\lambda+\phi)^2}\right) - \left(\frac{1}{(\lambda+\phi)^2} + \frac{1}{\lambda}\right)^2\right]^2,$$

$$\begin{aligned} N_1 = {} & \left(\frac{1}{(\lambda+\phi)^2} - \psi'(\phi)\right)\left[\left(\frac{1}{(\lambda+\phi)^2} - \psi'(\phi)\right)\left(\frac{\phi+1}{\lambda^3} - \frac{1}{(\lambda+\phi)^3}\right) + \left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{1}{2\lambda^2} + \frac{1}{(\lambda+\phi)^3}\right)\right] \\ & + 2\left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left[\left(\frac{1}{(\lambda+\phi)^2} - \psi'(\phi)\right)\left(\frac{1}{2\lambda^2} + \frac{1}{(\lambda+\phi)^3}\right) - \left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\frac{1}{(\lambda+\phi)^3}\right] \\ & + \left(\frac{1}{(\lambda+\phi)^2} - \frac{\phi+1}{\lambda^2}\right)\left[\left(\psi'(\phi) - \frac{1}{(\lambda+\phi)^2}\right)\frac{1}{(\lambda+\phi)^3} + \left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{1}{(\lambda+\phi)^3} + \frac{\psi''(\phi)}{2}\right)\right], \end{aligned}$$

$$\begin{aligned} N_2 = {} & \left(\frac{1}{(\lambda+\phi)^2} - \psi'(\phi)\right)\left[\left(\frac{\phi+1}{\lambda^2} - \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{1}{2\lambda^2} + \frac{1}{(\lambda+\phi)^3}\right) + \left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{1}{(\lambda+\phi)^3} - \frac{\phi+1}{\lambda^3}\right)\right] \\ & - 2\left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left[\left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\left(\frac{1}{2\lambda^2} + \frac{1}{(\lambda+\phi)^3}\right) + \left(\frac{\phi+1}{\lambda^2} - \frac{1}{(\lambda+\phi)^2}\right)\frac{1}{(\lambda+\phi)^3}\right] \\ & + \left(\frac{\phi+1}{\lambda^2} - \frac{1}{(\lambda+\phi)^2}\right)\left[\left(\frac{1}{(\lambda+\phi)^2} - \frac{\phi+1}{\lambda^2}\right)\left(\frac{1}{(\lambda+\phi)^3} + \frac{\psi''(\phi)}{2}\right) - \left(\frac{1}{\lambda} + \frac{1}{(\lambda+\phi)^2}\right)\frac{1}{(\lambda+\phi)^3}\right]. \end{aligned}$$

- Firth's method
  The estimator corrected by Firth's method is obtained by solving modified likelihood equations given by

$$\begin{pmatrix} \partial l/\partial\lambda \\ \partial l/\partial\phi \end{pmatrix} - A\,\text{vec}\left(K^{-1}\right) = \mathbf{0},$$

where

$$A = n\begin{bmatrix} \frac{\phi+1}{\lambda^3} - \frac{1}{(\lambda+\phi)^3} & -\frac{1}{2\lambda^2} - \frac{1}{(\lambda+\phi)^3} & -\frac{1}{2\lambda^2} - \frac{1}{(\lambda+\phi)^3} & -\frac{1}{(\lambda+\phi)^3} \\ -\frac{1}{2\lambda^2} - \frac{1}{(\lambda+\phi)^3} & -\frac{1}{(\lambda+\phi)^3} & -\frac{1}{(\lambda+\phi)^3} & -\frac{1}{(\lambda+\phi)^3} - \frac{1}{2}\psi''(\phi) \end{bmatrix},$$

$$K = n\begin{bmatrix} \frac{\phi+1}{\lambda^2} - \frac{1}{(\lambda+\phi)^2} & -\frac{1}{\lambda} - \frac{1}{(\lambda+\phi)^2} \\ -\frac{1}{\lambda} - \frac{1}{(\lambda+\phi)^2} & -\frac{1}{(\lambda+\phi)^2} + \psi'(\phi) \end{bmatrix}.$$

- Bootstrap method

$$\widehat{\boldsymbol{\theta}}_{BOOT} = \widehat{\boldsymbol{\theta}} - \widehat{\text{Bias}} = 2\widehat{\boldsymbol{\theta}} - \frac{1}{B}\sum_{b}^{B}\widehat{\boldsymbol{\theta}}^{*b},$$

where $\widehat{\boldsymbol{\theta}}^{*b}$s are bootstrap replications.

For bias correction, the '`bias_cor`' argument should be specified since the default is `bias_cor=NULL`. Note, unlike Cox and Snell's method that works with both MLE and MLE_c, Firth's method and the bootstrap method are only applicable to MLE and MLE_c, respectively. If other estimators are chosen with these correction methods, a default setting would be automatically adopted. The number of bootstrap iterations can be specified with `boot_iter`, while the default is 1000.

```
WL(fail_fiber, est_method = "MLE",bias_cor = "firth")   # Firth's method
```

```
WL(fail_fiber, est_method = "MLEc",bias_cor = "coxsnell") # Cox and Snell's method

WL(fail_fiber, est_method = "MLEc",bias_cor = "boots") # The bootstrap method
```

5 Interval estimation

Asymptotic confidence intervals

MLE, MLE_c, MME, and MME_m for the parameters of WL distribution achieve consistency and asymptotic normality. Thus, if the sample size is large enough, confidence intervals can be straightforwardly obtained with estimated asymptotic variances.

$$\widehat{\lambda} \pm z_{\alpha/2} \cdot \frac{\widehat{\sigma_1}}{\sqrt{n}} \quad \text{and} \quad \widehat{\phi} \pm z_{\alpha/2} \cdot \frac{\widehat{\sigma_2}}{\sqrt{n}}$$

One problem is that the lower bound of estimated intervals can be negative. A confidence interval with negative lower bound may not be useful since the parameters for WL distribution must be positive. Therefore, in case of negative lower bounds, confidence intervals for $\log(\boldsymbol{\theta})$ are first obtained by the delta method and then scaled back. That is,

$$\widehat{\lambda} \cdot \exp\left(\pm z_{\alpha/2} \cdot \frac{\widehat{\sigma_1}}{\sqrt{n} \cdot \widehat{\lambda}}\right) \quad \text{and} \quad \widehat{\phi} \cdot \exp\left(\pm z_{\alpha/2} \cdot \frac{\widehat{\sigma_2}}{\sqrt{n} \cdot \widehat{\phi}}\right).$$

Bootstrap confidence intervals

When asymptotic distribution is not available or the sample size is insufficient, the bootstrap confidence interval can be a good alternative. The basic bootstrap confidence interval is simply given by

$$\left(2\widehat{\lambda} - \widehat{\lambda}^*_{(1-\alpha/2)} \quad , \quad 2\widehat{\lambda} - \widehat{\lambda}^*_{(\alpha/2)}\right) \quad \text{and} \quad \left(2\widehat{\phi} - \widehat{\phi}^*_{(1-\alpha/2)} \quad , \quad 2\widehat{\phi} - \widehat{\phi}^*_{(\alpha/2)}\right),$$

where $\hat{\theta}^*_{(p)}$ denotes $(100 \cdot p)$ % percentile of bootstrap realizations. Readers are referred to Chapter 5 of **?** for more details. As in the asymptotic case, if the lower confidence limit is negative, confidence interval for $\log(\boldsymbol{\theta})$ is first computed and scaled back.

The default arguments of 'WL' function for interval estimation are `CI_method="asymp"`, `CI_scale="normal"`, `CI_side="two"`, and `CI_alpha=0.05`. If any bias correction method is used, `CI_method="boot"` is automatically chosen. `CI_scale="exp"` is recommended in the case of a negative lower confidence limit and one-sided intervals can be obtained by choosing `CI_side="one"`.

```
WL(fail_fiber,est_method="MLEc")$CI_list  #asymptotic CI for MLEc
WL(fail_fiber,est_method="MLE")$CI_list  #asymptotic CI for MLE

# Bootstrap CI for MLEc corrected by Cox and Snell's method
WL(fail_fiber,est_method = "MLEc", bias_cor = "coxsnell")$CI_list
```

6 Likelihood ratio test (LRT)

Let $X_1, \cdots, X_n$ be a random sample from $f(x \mid \theta), \boldsymbol{\theta} \in \Theta$, where $\boldsymbol{\theta} = (\lambda, \phi)^t$, $\Theta \subset \mathbb{R}^+ \times \mathbb{R}^+$ and $f(\boldsymbol{x} \mid \theta)$ is the pdf of WL distribution. To test $H_0 : \boldsymbol{\theta} \in \Theta_0$ vs. $H_1 : \boldsymbol{\theta} \in \Theta_0^c$, we reject H_0 if

$$2\log(R_n) = 2\log\left(\frac{\sup_\Theta l_n(\boldsymbol{\theta} \mid \boldsymbol{x})}{\sup_{\Theta_0} l_n(\boldsymbol{\theta} \mid \boldsymbol{x})}\right) > \chi^2_\alpha(2), \text{ where } l_n(\boldsymbol{\theta} \mid \boldsymbol{x}) = \prod_i f(x_i \mid \boldsymbol{\theta}).$$

The LRT is automatically implemented in the 'WL' function unless `wilks_test="FALSE"` is selected. The significance level and types of null hypotheses are set by `wilks_alpha` and `wilks_side`. The default setting is `wilks_alpha=0.05` and `wilks_side="two"`.

The LRT can be separately conducted with the `wilks.test` function apart from other inferences. This function is especially useful when testing several possible values or regions. By specifying arguments `estimator` and `side`, both simple and composite null hypotheses can be tested.

- H_0: $(\lambda, \phi) = (\lambda_0, \phi_0)$ vs H_1: not H_0.

```
wilks.test(fail_fiber,estimator = MME_WL(fail_fiber),side = "two")
```

- H_0: $\lambda \geq \lambda_0$ and $\phi \geq \phi_0$ vs H_1: not H_0.

```
wilks.test(fail_fiber,estimator=c(1,1),side="less")
```

7 Illustration - 'WL' function

In this section, we provide an example for illustrating how to conduct the aforementioned steps of statistical inference from the beginning using the 'WL' function. The 'WL' function returns an S3 object of class 'WL' which is assigned to `fiber` in this example. For this object of class 'WL', `summary` and `plot` functions are provided.

```
> data("fail_fiber")
> fiber = WL(fail_fiber,dist_test = "ks", est_method ="MLEc",
             wilks_alpha=0.05, wilks_side="two")
> summary(fiber)

Data: fail_fiber

Data summary:
Mean: 2.244, Variance: 0.1728
Min    1st Qu Median 3rd Qu Max
1.339  1.931  2.272  2.558  3.174

Kolmogorov-Smirnov test for alpha=0.05
D = 0.0723, p-value: 0.8864
>> This data follows the weighted Lindley distribution with estimated parameters.

Estimation method (bias correction): MLEc(None)
lambda: 12.9318, phi: 28.3329

Variance of lambda & phi:
Var(lambda)= 5.1126, Var(phi)= 25.2938

Two-sided confidence interval for 95%
(Asymptotic method & Normal scaled )
CI for lambda: (8.5001, 17.3635)
CI for phi: (18.4757, 38.1902)

Two-sided Wilks' theorem test for estimated parameters
X = 0.0024, p-value: 0.9988
>>The null hypothesis cannot be rejected.
```

The simple descriptive statistic is provided by '`summary(fiber)`', followed by the result of the goodness of fit test. This data set can be assumed to originate from WL distribution since the null hypothesis of the KS test is not rejected. We chose MLE_c without bias correction since the sample size of 65 is moderate, which allows us to use asymptotic variance to construct the confidence interval. `Variance of lambda & phi` in the output means estimated asymptotic variances obtained by replacing (λ, ϕ) with $\left(\widehat{\lambda}, \widehat{\phi}\right)$ in the variance formulas. Finally, the result of LRT with $H_0: \lambda = \widehat{\lambda}_{MLE_c}$ and $\phi = \widehat{\phi}_{MLE_c}$ is given.

To obtain some helpful plots, we use `plot` function. Four plots are returned by `plot(fiber)` as per Figure 1: the histogram with estimated density function, the boxplot for detecting outliers, QQ plot, and contour plot with point estimates. The first three plots provide insight on how much WL distribution is suitable for the given data, along with the goodness of fit tests. The last contour plot provides the surface of log likelihood near point estimates. Note that the density function in the histogram and the quantiles for the QQ plot are calculated based on estimated parameters.

8 Summary

We presented an R package **WLinfer** that implements a goodness of fit test, several types of point estimation, bias correction, interval estimation, and the likelihood ratio test, and provide some useful

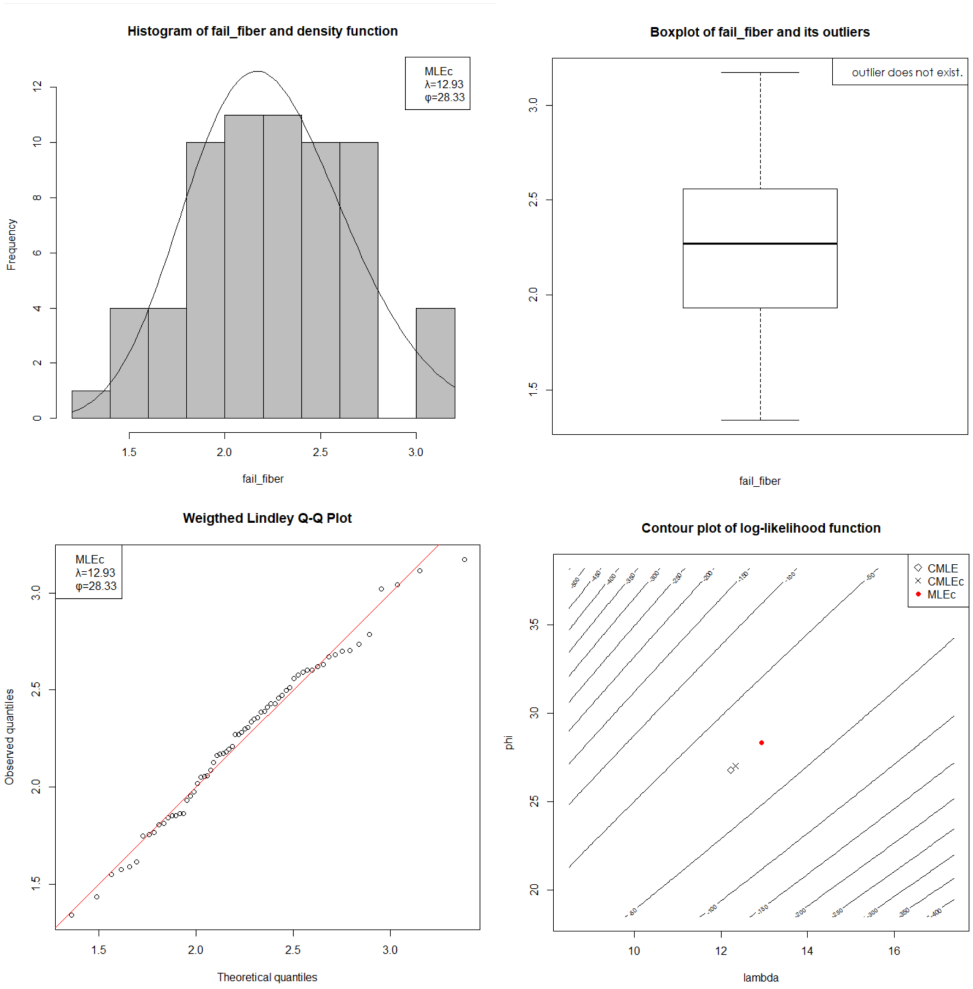


Figure 1: Histogram with estimated density function, boxplot for detecting outliers, weighted Lindley QQ plot, and contour plot of log-likelihood function with point estimates for a real data fail_fiber.

plots. We supply a set of simple codes and an illustrative example of how to apply this package in practice. This package could practically assist practitioners, removing the need to make codes from scratch.

9 Acknowledgments

The corresponding author's research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(2018R1D1A1B07045603) and a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2021R1A4A5032622).

Bibliography

M. Bader and A. Priest. Statistical aspects of fibre and bundle strength in hybrid composites. *in Progress in Science and Engineering of Composites, T. Hayashi , K. Kawata and S. Umekawa , eds., ICCM-IV, Tokyo*, 1982. [p15]

G. M. Cordeiro and R. Klein. Bias correction in arma models. *Statistics and Probability Letters*, 19(3):169 – 176, 1994. [p15]

D. R. Cox and E. J. Snell. A general definition of residuals. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(2):248–275, 1968. [p15]

M. Delignette-Muller and C. Dutang. fitdistrplus: An r package for fitting distributions. *Journal of Statistical Software, Articles*, 64(4):1–34, 2015. [p13]

B. Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, (7):1–26, 1979. [p15]

B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–75, 02 1986. [p15]

D. Firth. Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1):27–38, 03 1993. [p15]

M. Ghitany, F. Alqallaf, D. Al-Mutairi, and H. Husain. A two-parameter weighted lindley distribution and its applications to survival data. *Mathematics and Computers in Simulation*, 81(6):1190 – 1201, 2011. [p13]

M. E. Ghitany, P. Song, and S. Wang. New modified moment estimators for the two-parameter weighted lindley distribution. *Journal of Statistical Computation and Simulation*, 87(16):3225–3240, 2017. URL https://doi.org/10.1080/00949655.2017.1363205. [p14]

H.-M. Kim and Y.-H. Jang. New closed-form estimators for weighted lindley distribution. *submitted*, 2020. [p14, 15]

J. Mazucheli, F. Louzada, and M. Ghitany. Comparison of estimation methods for the parameters of the weighted lindley distribution. *Applied Mathematics and Computation*, 220:463 – 471, 2013. [p13, 14, 15]

J. Mazucheli, L. Fernandes, and R. de Oliveira. Package **LindleyR**. 2016. URL https://cran.r-project.org/web/packages/LindleyR/LindleyR.pdf. [p13]

J. Mazucheli, A. Menezes, and S. Nadarajah. mle.tools: An r package for maximum likelihood bias correction. *The R Journal*, 9(2):268–290, 2017. [p13]

M. Wang and W. Wang. Bias-corrected maximum likelihood estimation of the parameters of the weighted lindley distribution. *Communications in Statistics - Simulation and Computation*, 46(1): 530–545, 2017. URL https://doi.org/10.1080/03610918.2014.970696. [p15]

Yu-Hyeong Jang
Korea University
Department of Statistics
Seoul, Republic of Korea
jyhmaru@gmail.com

SungBum Kim
Konkuk University
Department of Applied Statistics
Seoul, Republic of Korea
kimsb0707@hanmail.net

Hyun-Ju Jung
Konkuk University
Department of Applied Statistics
Seoul, Republic of Korea
jhjstat@hanmail.net

Hyoung-Moon Kim (Corresponding author)
Konkuk University
Department of Applied Statistics
Seoul, Republic of Korea
(https://orcid.org/0000-0002-6014-5207)
hmkim@konkuk.ac.kr

ICAOD: An R Package for Finding Optimal designs for Nonlinear Statistical Models by Imperialist Competitive Algorithm

by Ehsan Masoudi, Heinz Holling, Weng Kee Wong and Seongho Kim

Abstract Optimal design ideas are increasingly used in different disciplines to rein in experimental costs. Given a nonlinear statistical model and a design criterion, optimal designs determine the number of experimental points to observe the responses, the design points and the number of replications at each design point. Currently, there are very few free and effective computing tools for finding different types of optimal designs for a general nonlinear model, especially when the criterion is not differentiable. We introduce an R package **ICAOD** to find various types of optimal designs and they include locally, minimax and Bayesian optimal designs for different nonlinear statistical models. Our main computational tool is a novel metaheuristic algorithm called imperialist competitive algorithm (ICA) and inspired by socio-political behavior of humans and colonialism. We demonstrate its capability and effectiveness using several applications. The package also includes several theory-based tools to assess optimality of a generated design when the criterion is a convex function of the design.

1 Introduction

Optimal designs have been extensively applied in many research studies to reduce the cost of experimentation. For instance, Holling and Schwabe (2013) provided examples in psychology and Dette et al. (2010) gave examples in dose-response studies. Further applications of optimal designs in engineering and epidemiology are described in Berger and Wong (2009), which also contains applications of optimal design ideas in other disciplines. Given a statistical model and an optimality criterion, optimal designs determine the optimal number of design points required, their locations to observe the responses and the number of replications required at each location. The optimality criterion should accurately reflect the objective of the study to the extent possible and is usually formulated as a scalar function of the Fisher information matrix (FIM) that measures the worth of the design (Lehmann and Casella, 1998). For example, if the objective of a study is to estimate the model parameters as accurately as possible, D-optimality is often used. Such an optimal design maximizes the determinant of the FIM and is called D-optimal. When errors are independent and normally distributed, D-optimal designs minimize the volume of the confidence ellipsoid of the model parameters by minimizing the generalized variance, i.e., the determinant of the variance-covariance matrix (Abdelbasit and Plackett, 1983).

For nonlinear models, the FIM depends on the unknown model parameters to be estimated and so the design criterion cannot be directly optimized. There are different approaches to deal with this parameter dependency: a) *locally optimal designs:* These are found by replacing the unknown parameters with some estimates obtained from a pilot or previous study (Chernoff, 1953). Locally optimal designs usually become inefficient when the replaced estimates are far from their true unknown values. b) *minimax optimal designs:* They minimize the maximum inefficiency over a user-selected parameter space (Sitter, 1992). The optimal designs are conservative in that they protect the experiment from the worst case scenario that may happen from a poor choice of parameter values over a user-specified space of plausible values for the unknown parameters. Finding minimax optimal designs is complicated because it involves solving multi-level nested optimization problems and the objective function (minimax criterion) is not differentiable. c) *Bayesian optimal designs:* These optimal designs are found by optimizing an optimality criterion averaged over a user-specified (continuous) prior distribution for the unknown parameters (Chaloner and Larntz, 1989; Chaloner and Verdinelli, 1995; Atkinson, 1996). Strictly speaking, the latter are not fully Bayesian because they do not involve computing a posterior distribution. Instead, they borrow the concept of having prior distributions to find robust designs for the frequentists (Graßhoff et al., 2012; Bürkner et al., 2019). Accordingly, they are sometimes referred to as "pseudo" Bayesian designs (Firth and Hinde, 1997). In the optimal design literature, Bayesian optimal designs found under a discrete prior distribution are usually referred to as *robust* or *optimum-on-average* designs (Fedorov and Hackl, 2012). For an overview of optimal designs for nonlinear models, see Fedorov and Leonov (2013).

There are several software packages to create and analyze design of experiment (DoE) for different

purposes. For a review on statistical R packages in design of experiments, see https://cran.r-project.org/web/views/ExperimentalDesign.html. Only a few of them are able to find different types of optimal designs to deal with the parameter dependency for various nonlinear models. To the best of our knowledge, none of the available software packages, commercial or otherwise, provides an option to find minimax optimal designs for nonlinear models. For example, the R package **LDOD** (Masoudi et al., 2013) finds locally $D-$optimal *approximate* designs for a large class of nonlinear models and the **acebayes** R package (Overstall et al., 2017) determines a more general class of fully Bayesian *exact* designs using the approximate coordinate exchange algorithm (Overstall and Woods, 2017). Likewise, the recently available **VNM** R package finds multiple-objective locally optimal designs for a specific model, i.e., the four-parameter Hill model commonly used in dose-response studies (Hyun et al., 2018). Among the commercial software, JMP® (SAS Institute Inc., 2016) can also find Bayesian $D-$optimal exact designs for nonlinear models.

This paper introduces the R package **ICAOD** (Masoudi et al., 2020) for finding a variety of optimal designs for nonlinear models using a novel metaheuristic algorithm called *imperialist competitive algorithm(ICA)*. This algorithm is inspired by socio-political behavior of humans (Atashpaz-Gargari and Lucas, 2007; Hosseini and Al Khaled, 2014) and is modified by Masoudi et al. (2017) and Masoudi et al. (2019) to find optimal designs for nonlinear models. We believe that this **ICAOD** package is the first single self-contained statistical package that presents a framework to find locally, minimax and Bayesian optimal designs for nonlinear models. Similar to many popular nature-inspired metaheuristic algorithms, such as particle swarm optimization (PSO) algorithm (Kennedy and Eberhart, 1995), ICA does not have a rigorous proof of convergence (Yang, 2011). When the criterion is a convex function on the set of design measures, equivalence theorems are available and the **ICAOD** package includes tools to confirm optimality of a design. More generally, the proximity of any design to the optimum without knowing the latter can be evaluated in terms of an efficiency lower bound. In particular, if this bound is unity, this confirms optimality of the design. This feature is useful to recognize a case of pre-mature convergence in optimal design problems.

The next section reviews the statistical setup and theory for finding optimal designs for nonlinear models. The fourth section describes the imperialist competitive algorithm (ICA) and the fifth section provides implementation details for the **ICAOD** package. In the sixth section, we provide two examples to show the functionality of the **ICAOD** package. The seventh section finds locally and minimax $D-$optimal designs for a logistic model with application in educational testing and The eighth section presents optimum-on-average and Bayesian $D-$optimal designs for a sigmoid Emax model for dose-response studies. The **ICAOD** package was first written to find locally $D-$optimal designs, but it now also finds user-defined optimal designs. The ninth section illustrates how to use this feature to find $c-$optimal designs for a two-parameter logistic model in dose response studies. The last section concludes with a summary.

2 Background and optimal designs

Let $E(Y) = f(\boldsymbol{x}, \boldsymbol{\theta})$ be the mean of the response Y at the values of the independent variables $\boldsymbol{x}$ defined on a user-selected *design space* χ, and let f be a known function, apart from the model parameters $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)^T$. Throughout we assume that there are resources to take N observations for the study and given an optimality criterion, we want to find the best choices for the levels of the independent variables to observe the outcome Y. There are two types of designs: exact and approximate.An *exact* design ξ_N on χ is defined by a set of k distinct levels $\boldsymbol{x}_i$,

$$\xi_N = \left\{ \begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & ... & \boldsymbol{x}_k \\ n_1/N & n_2/N & ... & n_k/N \end{array} \right\}, \tag{1}$$

where $\boldsymbol{x}_j \in \chi$, n_j is the number of replications of $\boldsymbol{x}_j$ in the observations sample and $N = \sum_{j=1}^{k} n_j$. Here, $x_j, j = 1..., k$ are referred to as *support points* or *design points* of ξ_N. Given N and a specific design criterion, an optimal exact design finds the best value of k and the best values of $x_1, \ldots, x_k, n_1, \ldots, n_k$. Such optimization problems are notoriously difficult and in practice, we find optimal approximate designs instead. They are probability measure on χ are found independent of the sample size N. An approximate design ξ with k support points has the form

$$\xi = \left\{ \begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & ... & \boldsymbol{x}_k \\ w_1 & w_2 & ... & w_k \end{array} \right\}, \tag{2}$$

where $w_j > 0$ is the proportion of observations that is assigned to $\boldsymbol{x}_j$ and $\sum_{j=1}^{k} w_j = 1$. It is implemented by first rounding each value of Nw_i to the nearest integer Nw_i^* subject to $Nw_1^* + \ldots + Nw_k^* = N$ and taking Nw_i* observations at $\boldsymbol{x}_i, i = 1, \ldots, k$. Some optimal rounding procedures are

available in Pukelsheim and Rieder (1992). When the design criterion is formulated as a convex function of the FIM, there are algorithms for finding many types of optimal approximate designs and theory to confirm optimality of an approximate design. When the design is not optimal, a theory-based efficiency lower bound of the design is available to determine its proximity to the optimum, without knowing the optimum. For these reasons, we focus on optimal approximate designs found under a convex functional in the rest of the paper.

To find an approximate design that minimizes a convex design criterion ψ over the space of all designs on χ. We have to determine the optimal number of support points, k, the optimal support points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ and their corresponding $w_1, \ldots, w_k$. For example, if estimating model parameters is of interest, $D-$optimality, defined by the logarithm of the determinant of the inverse of the FIM, is a convex functional over the space of all designs on χ (Fedorov and Leonov, 2013; Silvey, 1980) and the design that minimuzes it is called $D-$optimal. In what follows, we focus on the $D-$optimality criterion and briefly discuss other optimality criteria and optimal designs which can be studied similarly.

Assuming all observation errors are independent and normally distributed with means 0 and a constant variance (Y), the FIM of a generic k-point approximate design ξ is given by

$$M(\xi, \boldsymbol{\theta}) = \sum_{i=1}^{k} w_i I(\boldsymbol{x}_i, \boldsymbol{\theta}), \tag{3}$$

where

$$I(\boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{1}{(Y_i)} \nabla f(\boldsymbol{x}_i, \boldsymbol{\theta}) \nabla f(\boldsymbol{x}_i, \boldsymbol{\theta})^T,$$

and $\nabla f(\boldsymbol{x}_i, \boldsymbol{\theta})^T = \left(\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_1}, \cdots, \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_p} \right)$. Here, $\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_j}$ denotes the partial derivative of f with respect to θ_j. The FIM is singular if $k < p$. To avoid singular designs, i.e., designs with singular Fisher information matrices, we assume $k \geq p$.

Clearly, the FIM (3) depends on the unknown parameters for nonlinear models. Different approaches have been proposed to deal with this parameter dependency based on the type of information available for the unknown parameters. For example, let $\boldsymbol{\theta}_0$ be an initial guess for $\boldsymbol{\theta}$ available from a similar study. A locally $D-$optimal design ξ^*_{loc} minimizes

$$\psi_{loc}(\xi) = -\log |M(\xi, \boldsymbol{\theta}_0)|, \tag{4}$$

where $|\cdot|$ denotes the determinant. In practice, it is more realistic to assume that the unknown parameters belong to a user-specified parameter space Θ, which is comprised of all possible values for $\boldsymbol{\theta}$. Given Θ, we can find minimax optimal designs that minimize the maximum inefficiency over Θ and protect the experiment from the worst-case scenario over the parameter space. A minimax $D-$optimal design ξ^*_{min} is obtained by minimizing

$$\psi_{min}(\xi) = \max_{\boldsymbol{\theta} \in \Theta} -\log |M(\xi, \boldsymbol{\theta})|, \tag{5}$$

over the space of all designs on χ. The minimax problem (5) is a bi-level nested optimization problem with inner and outer optimization problems. Given any arbitrary design, the inner optimization problem is to maximize the $D-$criterion $-\log |M(\xi, \boldsymbol{\theta})|$ over Θ to find the maximum inefficiency and the outer optimization problem is to minimize the maximum of the inner problem over the space of all designs on χ. Alternatively, when a prior distribution $\pi_\Theta(\boldsymbol{\theta})$ is available for the unknown parameters on Θ, Bayesian optimal designs may also be found: a (pseudo) Bayesian $D-$optimal design ξ^*_{bayes} minimizes

$$\psi_{bayes}(\xi) = \int_{\boldsymbol{\theta} \in \Theta} -\log |M(\xi, \boldsymbol{\theta})| \pi_\Theta(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{6}$$

When $\pi_\Theta(\boldsymbol{\theta})$ is a discrete prior, the obtained designs are sometimes referred to as *optimum-on-average* or *robust* designs.

One advantage of working with approximate designs is existence of an equivalence theorem, which can be used to verify the optimality of a given design if the criterion is a convex function on the set of design measures. Each convex optimality criterion gives rise to a different equivalence theorem, but they generally have the same form. For example, a design ξ^*_{loc} is locally $D-$optimal if and only if the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{loc}(\boldsymbol{x}, \xi^*_{loc}) = \operatorname{tr} M^{-1}(\xi^*_{loc}, \boldsymbol{\theta}_0) I(\boldsymbol{x}, \boldsymbol{\theta}_0) - p \leq 0, \tag{7}$$

with equality in (7) at all support points of ξ^*_{loc}. The left hand-side of inequality (7) is sometimes

called *sensitivity function*. The equivalence theorem for Bayesian $D-$optimality criterion is very similar (Kiefer and Wolfowitz, 1959; Chaloner and Larntz, 1989): a design ξ^*_{bayes} is a Bayesian $D-$optimal design if and only if the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{bayes}(\boldsymbol{x}, \xi^*_{bayes}) = \int_{\Theta} \text{tr}\{M^{-1}(\xi^*_{bayes}, \boldsymbol{\theta}) I(\boldsymbol{x}, \boldsymbol{\theta})\} \pi(\boldsymbol{\theta}) d\theta - p \leq 0, \tag{8}$$

with equality in (8) at all support points of ξ^*_{bayes}. However, the equivalence theorem for a minimax type criterion takes on a more complicated form because (5) is not differentiable. The equivalence theorem states that a design ξ^*_{min} is minimax $D-$optimal among all the designs on χ if and only if there exists a probability measure μ^* on

$$A(\xi^*_{min}) = \left\{ \boldsymbol{\nu} \in \Theta \mid -\log|M(\xi^*_{min}, \boldsymbol{\nu})| = \max_{\boldsymbol{\theta} \in \Theta} -\log|M(\xi^*_{min}, \boldsymbol{\theta})| \right\}, \tag{9}$$

such that the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{min}(\boldsymbol{x}, \xi^*_{min}) = \int_{A(\xi^*_{min})} \text{tr}\, M^{-1}(\xi^*_{min}, \boldsymbol{\nu}) I(\boldsymbol{x}, \boldsymbol{\nu}) \mu^* d(\boldsymbol{\nu}) - p \leq 0, \tag{10}$$

with equality in (10) at all support points of ξ^*_{min} (Wong, 1992; Fedorov, 1980; King and Wong, 2000; Berger et al., 2000). The set $A(\xi^*_{min})$ is sometimes called the *answering set* of ξ^* and the measure μ^* is a sub-gradient of the non-differentiable criterion evaluated at $M(\xi^*_{min}, \nu)$. Understanding the properties of the sub-gradients and how to find them efficiently for the minimax optimal design problems present a key problem in solving this type of problems. In particular, there is no theoretical rule on how to choose the number of points in $A(\xi^*_{min})$ as support for the measure μ^* and they would have to be found by trial-and-error. For more details, see Masoudi et al. (2017). When χ is one or two dimensional, it is very common to plot the sensitivity function versus $\boldsymbol{x} \in \chi$ and visually inspect whether the graph meets the conditions in the equivalence theorem. If it does, the generated design is optimal; otherwise it is not optimal.

We measure the efficiency of one design ξ_1 relative to another design ξ_2 using their criterion values. For example, for $D-$optimality (4), we use

$$\text{eff}_{loc} = \left(\frac{|M(\xi_1, \boldsymbol{\theta})|}{|M(\xi_2, \boldsymbol{\theta})|} \right)^{1/p} = \exp\left(\frac{\psi_{loc}(\xi_2) - \psi_{loc}(\xi_1)}{p} \right). \tag{11}$$

The relative $D-$efficiency (11) may be interpreted in term of sample size; if its value is ρ, then ξ_1 requires $1/\rho$ times as many observations to have the same $D-$efficiency as ξ_2. This means that, when ξ_2 is an optimal design, about $(1/\rho - 1)100\%$ more number of observations are required for design ξ_1 to do as well as the optimal design. Similarly, we define Bayesian and minimax $D-$efficiencies by replacing ψ_{loc} with ψ_{min} and ψ_{bayes}, respectively. Standardly, (11) becomes the $D-$efficiency of ξ_1 when ξ_2 is the $D-$optimal design.

When the design criterion is a convex functional, we can use the equivalence theorem to quantify the proximity of a design ξ to the optimal design without knowing the latter by means of the efficiency lower bound (ELB). For example, for $D-$optimality, we have

$$\text{ELB} = \frac{p}{p + \max_{\boldsymbol{x} \in \chi} c(\boldsymbol{x}, \xi)}, \tag{12}$$

where $c(\boldsymbol{x}, \xi)$ is the sensitivity function associated with $D-$optimality. The value of the ratio in (12) is between 0 and 1, and it is equal to 1 if and only if the design is optimal. The efficiency bounds are not unique and can be varously derived using somewhat similar arguments, for example, see Atwood (1969) and Pázman (1986).

3 Imperialist competitive algorithm for finding optimal designs

The imperialist competitive algorithm (ICA) is an evolutionary algorithm inspired from colonialism and socio-political behavior of humans, where developed countries attempt to take over or colonize less-developed countries to use their resources and extend their power (Atashpaz-Gargari and Lucas, 2007). Within the optimization framework, ICA has a population of solutions called *countries*. In optimal design problems, each country is the location of the support points and the corresponding weights of a design on the space of all possible designs. ICA divides the population of countries into some sub-populations called *empires*. Each empire contains one *imperialist* and some *colonies*. The imperialist is the most powerful country within the empire. Here, the power of a country is

defined to be a function of its cost value, i.e., criterion value. This means that, in a minimization problem, countries with smaller cost values are stronger. In ICA, there are two types of evolutionary moves: a) evolution within each empire, and, b) evolution among the empires. In the earlier, the colonies within each empire start to move or be absorbed toward their relevant imperialist country in a process called *assimilation* (Lin et al., 2013). During this process, a colony may reach a better position than its imperialist. In this case, the imperialist loses its rank and the colony becomes the new imperialist. The assimilation improves searching around the better current solutions and so enhances the exploitation of the algorithm.

The evolution among the empires is achieved by a process called *imperialists competition*. In this process, the most powerful empires receive more chances to take possession of the colonies of the weakest empires. The competition step in ICA improves the exploration of the algorithm in a search for the global optimum. When an empire does not have any colony, it will be eliminated. ICA continues until it satisfies the stopping rule conditions. For more details, see Atashpaz-Gargari and Lucas (2007) and Hosseini and Al Khaled (2014).

To apply ICA for an optimal design problem, the user should first provide an initial guess about the number of support point $k(\geq p)$. In practice, the user can start by p and increment its value by one until the equivalence theorem confirms the optimality of the current best design, which is the country with the least cost value. In optimal design problems, the ELB defined by (12) can be used to build a stopping rule condition for ICA. For example, the algorithm can be stopped when the value of the ELB of the best current design is larger than, say, 0.95. Clearly, finding ELB in each iteration increases the CPU time required by the algorithm as another optimization problem has to be solved to find maximum of the sensitivity function over χ. This is especially true for minimax and Bayesian type criteria, because the sensitivity function for the earlier involves solving a bi-level nested optimization problem and the latter requires approximating integrals. Therefore, we prefer to calculate the ELB periodically, say, after every 100 iterations, instead of every iteration to save the CPU time.

4 Implementation of optimal design problems in ICAOD

Different functions are available to find optimal designs for nonlinear models in **ICAOD**: a) `locally()`: Finds locally optimal designs, b) `minimax()`: Finds minimax optimal designs , c) `bayes()`: Finds Bayesian optimal designs and d) `robust()`: Finds optimum-on-average or robust designs. Throughout this paper, we refer to them as "OD functions". **ICAOD** uses the S3 object oriented system and works with an object of class '`minimax`'. The class '`minimax`' has its own `plot`, `print` and `update` method. The `plot` method is used to plot the sensitivity function and also calculate the ELB for the output design. The `print` method is to display the brief profile of ICA iterations and the summary of identified optimal designs. The `update` method is for executing the algorithm for more number of iterations. By default, OD functions are defined to determine $D-$optimal designs. In the section '`User-Specified Optimality Criteria`', we demonstrate how to specify user-defined optimality criteria. In what follows, the OD functions are explained in detail.

Locally optimal designs

The `locally()` function finds locally optimal designs and its main arguments are:

```
locally(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
        lx, ux, k, iter, ICA.control = list(), sens.control = list(),
        crt_func = NULL, sens_func = NULL,
        inipars)
```

The arguments in the first three lines of codes are common between the OD functions. Table 1 provides an overview of them. The arguments in the first line are required to construct the FIM of the model; `inipars` is equivalent to $\boldsymbol{\theta}_0$ in (4) and defines the vector of initial estimates for the model parameters.

The **ICAOD** package includes a formula interface to specify the model of interest. For example, assume the two-parameter logistic model defined by

$$f(x, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-b(x - a))}, \tag{13}$$

where $\boldsymbol{\theta} = (a, b)$ is the vector of model parameters and x is the model predictor. To define (13) in **ICAOD**, we can set `formula = 1/(1 + exp(-b * (x-a)))`, `predvars = "x"`, `parvars = c("a","b")` and `family = "binomial"` (or `family = binomial()`). Alternatively, one may pass

Argument	Description
`formula`	A formula that is the symbolic description of a nonlinear model.
`predvars`	A vector of characters that denote the model predictors in `formula`.
`parvars`	A vector of characters that denote the model parameters in `formula`.
`family`	The distribution of the model response and the link function. It is the same as the one in `glm()`. The default link function is `gaussian()`.
`fimfunc`	(optional) The Fisher information matrix (R `function`). Required if users wish to pass the FIM directly. It takes a `function` with arguments `x` (a vector of design points), `w` (a vector of associated weights) and `param` (a vector of model parameters). Only one of the `formula` and `fimfunc` arguments must be given.
`k`	The number of design points k.
`lx`	A vector of the lower bounds for the model predictors (design space χ).
`ux`	A vector of the upper bounds for the model predictors (design space χ).
`x`	(optional) A vector of design points $\boldsymbol{x}$. if given, only the optimal weights, `w`, are sought after. Required when the design points are pre-specified.
`ICA.control`	A `list` of ICA control parameters. By default, it will be created by `ICA.control()`.
`iter`	The maximum number of iterations.
`sens.control`	Control Parameters of the maximization algorithm, which finds the maximum of the sensitivity function (7), (10) and (8) over the design space χ. The obtained maximum is used to calculate the ELB of a design. By default, it will be created by `sens.control()`.
`crt_func`	(optional) A user-specified criterion (R `function`).
`sens_func`	(optional) A user-specified sensitivity function (R `function`).

Table 1: Overview of the most important common arguments of the OD functions.

the FIM of (13) as an R `function` via the argument `fimfunc` directly. In this option, the arguments of the defined function must be a) `x`: is a `vector` of $(\boldsymbol{x}_1, ..., \boldsymbol{x}_k)$ in (2), b) `w`: is a `vector` of $(w_1, ..., w_k)$ in (2), and c) `param`: is a `vector` of $\boldsymbol{\theta}$ in (13). The output is the FIM of (13) evaluated at the given `x`, `w` and `param` as a `matrix`.

The argument `sens.control` is a list of control parameters for `nloptr()` available in the **nloptr** package (Johnson, 2014). This function is used here to solve $\max_{\boldsymbol{x}\in\chi} c(\boldsymbol{x}, \xi)$ for computing the ELB (12). When not given, it will be created automatically by the function `sens.control`. We recommend not to change its default values as they have been successfully tested for a large number of problems.

The `crt_func` and `sens_func` arguments are used to find a user-defined optimal designs, which are described in the section '`User-Specified Optimality Criteria`'.

Minimax optimal designs

The `minimax()` function finds minimax optimal designs and its main arguments are:

```
minimax(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
        lx, ux, k, iter, ICA.control = list(), sens.control = list(),
        crt_func = NULL, sens_func = NULL,
        lp, up, n.grid = 0,
        sens.minimax.control = list(), crt.minimax.control = list())
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to evaluate the minimax criterion (5) and its sensitivity function (10) at a given design. Table 2 presents an overview of the arguments specifically available in `minimax()`.

In **ICAOD**, the parameter space Θ are either continuous or discrete. Note that the lower bound and upper bound of Θ are specified via the arguments `lp` and `up`, respectively. When Θ is continuous, **ICAOD** uses `nloptr()` to solve the inner maximization problem in (5) over Θ at a given design. The default optimization algorithm from `nloptr()` is the DIRECT-L algorithm, which is a deterministic search algorithm based on the systematic division of the search domain into smaller and smaller hyperrectangles (Gablonsky and Kelley, 2001). For our applications, the most influential tuning parameter of `nloptr()` is the maximum number of function evaluations denoted by `maxeval` (its default value is 1000) via the `crt.minimax.control` argument. The parameter space may also be

Argument	function	Description
`lp`	`minimax()`	A vector of lower bounds for $\boldsymbol{\theta}$.
`up`		A vector of upper bounds for $\boldsymbol{\theta}$.
`n.grid`		(optional) When have a positive value, the parameters space Θ will be discretized, where the number of grid points will be equal to `n.grid^p` (defaults to 0).
`crt.minimax.control`		A list of control parameters of the function `nloptr()`, which is used to maximize the optimality criterion at a given design over Θ. By default, it will be created by `crt.minimax.control()`.
`sens.minimax.control`		A list of control parameters to find the answering set (9), which is required to obtain the sensitivity function and calculate the ELB. By default, it will be created by `sens.minimax.control()`. For more details, see `?sens.minimax.control`.
`prior`	`bayes()`	An object of class '`cprior`' that contains the necessary information about the prior distribution for the unknown parameters $\boldsymbol{\theta}$. For popular prior distributions, it can be created via the `uniform()`, `normal()`, `skewnormal()`, `student()` functions. For more details, see `?bayes`.
`crt.bayes.control`		A list of control parameters to approximate the integrals in (6), using either the `hcubature()` function (an adaptive multidimensional integration method over hypercubes) or the Gaussian quadrature formulas implemented by the **mvQuad** package. By default, it will be created by `crt.bayes.control()`.
`sens.bayes.control`		A list of control parameters required to approximate the integrals in (8). It is very similar to `crt.bayes.control()` and by default will be created by `crt.bayes.control()`.
`prob`	`robust()`	A `vector` of the probability measure associated with each vector of initial estimates for the unknown parameters $\boldsymbol{\theta}$.
`parset`		A `matrix` where each of its row is a vector of the initial estimates for $\boldsymbol{\theta}$.

Table 2: Overview of the arguments that are used to evaluate minimax, Bayesian and robust (optimum-on-average) optimality criteria at a given design.

discretized. In this option, the total number of grid points is equal to `n.grid^p`. When specified, ICA evaluates the criterion at these grid points to solve the maximization problem over Θ.

Bayesian optimal designs

The `bayes()` function finds Bayesian optimal designs and its main arguments are:

```
bayes(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
      lx, ux, k, iter, ICA.control = list(), sens.control = list(),
      crt_func = NULL, sens_func = NULL,
      prior, crt.bayes.control = list(), sens.bayes.control = list())
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to approximate the integrals in (6) and (8) at a given design. Table 2 presents an overview of the arguments specifically available in `bayes()`.

By default, **ICAOD** uses the `hcubature()` function from the **cubature** package (Johnson, 2013; Narasimhan and Johnson, 2017) to approximate the integrals. Th function `hcubature()` includes an adaptive multidimensional integration method over hypercubes known as hcubature algorithm

(Berntsen et al., 1991; Genz and Malik, 1980). For our applications, the most important tuning parameters of the hcubature algorithm are the maximum number of integrand evaluations `maxEval` (its default value is `50000`) and a user-specified tolerance `tol` (its default value is `1e-5`). This algorithm stops either when the integral error estimate is less than the integral estimate multiplied by its value or when the it reaches the specified maximum number of function evaluations `maxEval`, whichever happens earlier. When the prior distribution is less diffuse, it is sometimes more efficient to reduce the value of `maxEval` to increase the speed of the hcubature algorithm. The control parameters of the `hcubature()` function can be regulated via the argument `crt.bayes.control`.

Alternatively, **ICAOD** also offers the Gauss-Legendre and the Gauss-Hermite formulas to approximate the integrals. These methods are implemented in ICA using the **mvQuad** package (Weiser, 2016) and can be requested via the argument `crt.bayes.control`. For more details, see `?mvQuad::createNIGrid()`.

Robust or optimum-on-average designs

The `robust()` function finds optimum-on-average or robust designs and its main arguments are:

```
robust(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
       lx, ux, k, iter, ICA.control = list(), sens.control = list(),
       crt_func = NULL, sens_func = NULL,
       prob, parset)
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to evaluate the optimum-on-average criterion at a given design. Table 2 presents an overview of the arguments specifically available in `robust()`.

5 Examples

In this section, we provide two examples to show the functionality of the **ICAOD** package to determine optimal designs. In the first example, we find locally and minimax $D-$optimal designs for a logistic model with applications in educational testing. In the second example, we specify Bayesian and robust optimal designs for the sigmoid Emax model with applications in dose-response studies.

Logistic model with a single predictor

The logistic model is very popular for modeling binary outcomes. For example, consider an educational research that studies the effect of hours of practice on the mastery of a mathematical task. Let Y be a binary response variable that takes the value 1 if a subject has mastered the task and 0 otherwise. The logistic model is defined by

$$f(x,\boldsymbol{\theta}) = P(Y=1) = \frac{\exp(\beta_0+\beta_1 x)}{1+\exp(\beta_0+\beta_1 x)}, \tag{14}$$

where x is the hours of practice and $\boldsymbol{\theta} = (\beta_0,\beta_1)^T$. Assume that for each subject up to six hours of practice are possible, i.e., $x \in \chi = [0,6]$. If the purpose of the study is to estimate the model parameters accurately, an appropriate criterion is the $D-$optimality. The design questions here are a) what is the best number of levels of x to apply in the study, b) what are these levels and c) how many subjects should be assigned to each level? For example, a researcher may choose a uniform design that includes an equal number of subjects who have practiced for $0,1,2,3,4,5,6$ hours. We denote this design by

$$\xi_{uni} = \left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{array} \right\}. \tag{15}$$

The FIM of model (14) depends on the unknown parameters through $\frac{\partial f(\boldsymbol{x},\boldsymbol{\theta})}{\partial \beta_j}, j=0,1$. Following Berger and Wong (2005), let $\boldsymbol{\theta}_0 = (-4, 1.3333)^T$ be the best initial guess for $\boldsymbol{\theta}$ available from, say, a similar study. In **ICAOD**, the locally $D-$optimal design is found by

```
R> library("ICAOD")
R> log1 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40, k = 2,
+                  inipars = c(-4, 1.3333), ICA.control = list(rseed = 1))
```

```
R> print(log1)
Finding  locally optimal designs

Call:
  ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x))

iter       x1       x2        w1        w2 min_cost mean_cost
1     1 1.897630 4.289279 0.4480275 0.5519725 3.585707  3.629564
5     5 1.735791 4.135709 0.4818501 0.5181499 3.574277  3.608956
9     9 1.812077 4.132323 0.4965689 0.5034311 3.569134  3.569134
14   14 1.831070 4.143704 0.4982596 0.5017404 3.568777  3.568777
18   18 1.845842 4.158716 0.4996087 0.5003913 3.568684  3.568684
22   22 1.842875 4.159136 0.4999992 0.5000008 3.568680  3.568680
27   27 1.842477 4.157866 0.4999457 0.5000543 3.568679  3.568679
31   31 1.842450 4.157647 0.4999709 0.5000291 3.568679  3.568679
35   35 1.842456 4.157634 0.4999973 0.5000027 3.568679  3.568679
40   40 1.842479 4.157646 0.4999987 0.5000013 3.568679  3.568679

Optimal designs (k=2):
  Points1 Points2
1.84248 4.15765
Weights1 Weights2
0.500   0.500

ICA iteration: 40
Criterion value:  3.568679
Total number of function evaluations: 1768
Total number of successful local search moves: 76
Total number of successful revolution moves: 48
Convergence: Maximum_Iteration
Total number of successful assimilation moves: 701
CPU time: 1.09  seconds!
```

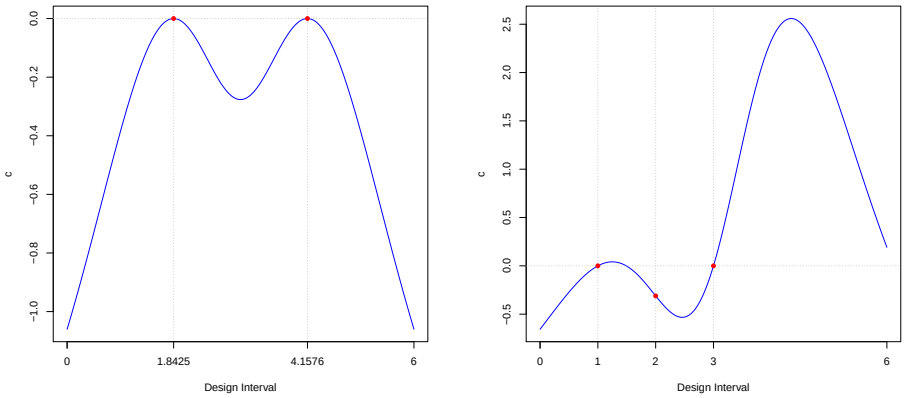
Throughout this paper, the `rseed` argument is used to guarantee the reproducibility of the results. The algorithm stopped at iteration number 40 because it reached the maximum number of iterations (`iter = 40`). Here, the design provided by the output assigns equal weights to $x_1 = 1.84249$ and 4.15765. This mean that, half of the subjects should be assigned to practice nearly less than 2 hours and the other half should practice a little bit more than 4 hours. The $D-$criterion (4) evaluated at this design is equal to 3.5686. Alternatively, the optimal design at the final iteration and the detailed profiles of ICA optimization at each iteration can be obtained by

```
R> log1$design
iter       x1       x2        w1        w2 min_cost mean_cost max_sens elb
1   40 1.842479 4.157646 0.4999987 0.5000013 3.568679  3.568679       NA  NA
time_sec
1       NA
R> log1$out
iter       x1       x2        w1        w2 min_cost mean_cost
1     1 1.897630 4.289279 0.4480275 0.5519725 3.585707  3.629564
2     2 1.894919 4.287451 0.4875810 0.5124190 3.575292  3.619014
3     3 1.895518 4.285989 0.4875810 0.5124190 3.575159  3.616086
4     4 1.735791 4.135761 0.4818501 0.5181499 3.574278  3.613881
5     5 1.735791 4.135709 0.4818501 0.5181499 3.574277  3.608956
...
36   36 1.842457 4.157634 0.4999973 0.5000027 3.568679  3.568679
37   37 1.842457 4.157634 0.4999973 0.5000027 3.568679  3.568679
38   38 1.842460 4.157632 0.4999980 0.5000020 3.568679  3.568679
39   39 1.842481 4.157638 0.5000023 0.4999977 3.568679  3.568679
40   40 1.842479 4.157646 0.4999987 0.5000013 3.568679  3.568679
```

The plot of the sensitivity function of the design provided by the output and the value of the ELB is obtained by

```
R> plot(log1)
Maximum of the sensitivity function is  5.323248e-06
Efficiency lower bound (ELB) is  0.9999973
```

```
Verification required 0.33 seconds!
```



(a) `Output from plot(log1)` **(b)** `Output from plot(log3)`

Figure 1: Plots of the sensitivity functions of the designs generated by the `locally()` function for the logistic model over $\chi = [0, 6]$ when $\boldsymbol{\theta} = \boldsymbol{\theta}_0 = (-4, 1.3333)^T$. The left panel (a) verrfies the global optimality of the obtained design and the right panel (b) does not verify the optimality of the obtained design. The solid red dots are the values of the sensitivity function at the obtained design points.

Figure 1 (a) displays the plot of the sensitivity function (7) of the design provided by the output on the design space $[0, 6]$. Based on the equivalence theorem, this design is optimal because the sensitivity function is equal or less than zero on $[0, 6]$ and (roughly) equal to zero at 1.84249 and 4.15765 (see the red points). The value of the ELB is nearly 1, which also indicates the optimality of this design.

It is interesting to assess the performance of the uniform design ξ_{uni} with respect to the locally $D-$optimal design obtained above. Using (11), we can calculate the $D-$efficiency of ξ_{uni} relative to the locally $D-$optimal design by

```
R> leff(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+      predvars = "x", parvars = c("b0", "b1"),
+      family = "binomial", inipars = c(-4, 1.3333),
+      x1 = c(0:6), w1 = rep(1/7, 7),
+      x2 = log1$evol[[20]]$x, w2 = log1$evol[[20]]$w)
[1] 0.7778719
```

The value of the relative $D-$efficiency indicates that ξ_{uni} requires about $100(1/0.777 - 1) = 29\%$ more number of subjects to have the same $D-$efficiency as the $D-$optimal design when $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. Therefore, having subjects to practice, say, less than 1 hours or more than 5 hours will not increase the efficiency of the parameter estimates very much.

The value of the ELB may also be used to construct a stopping rule condition for ICA. This feature is activated via the `ICA.control` argument in all OD functions similar to what follows.

```
R> log2 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40, k = 2,
+                  inipars = c(-4, 1.3333),
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 20,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99))
R> print(log2)
Finding  locally optimal designs

Call:
  ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x))

iter       x1       x2        w1        w2 min_cost mean_cost
```

```
1     1 1.897630 4.289279 0.4480275 0.5519725 3.585707  3.629564
3     3 1.895518 4.285989 0.4875810 0.5124190 3.575159  3.616086
5     5 1.735791 4.135709 0.4818501 0.5181499 3.574277  3.608956
7     7 1.818028 4.160041 0.4797011 0.5202989 3.570617  3.570617
9     9 1.812077 4.132323 0.4965689 0.5034311 3.569134  3.569134
11   11 1.827779 4.137145 0.4958561 0.5041439 3.568919  3.568919
13   13 1.844558 4.142393 0.4961667 0.5038333 3.568856  3.568856
15   15 1.845992 4.165776 0.4984264 0.5015736 3.568713  3.568713
17   17 1.845348 4.155565 0.4996783 0.5003217 3.568688  3.568688
20   20 1.842781 4.159234 0.4999992 0.5000008 3.568680  3.568680

Optimal designs (k=2):
  Points1 Points2
1.84278 4.15923
Weights1 Weights2
0.500   0.500

ICA iteration: 20
Criterion value:  3.56868
Total number of function evaluations: 918
Total number of successful local search moves: 46
Total number of successful revolution moves: 46
Convergence: equivalence
Total number of successful assimilation moves: 345
CPU time: 0.81  seconds!

  Maximum of the sensitivity function is  3.483904e-06
Efficiency lower bound (ELB) is  0.9999983
Verification required 0.39 seconds!
  R> log2$design
iter       x1       x2        w1        w2 min_cost mean_cost     max_sens
1   20 1.842781 4.159234 0.4999992 0.5000008  3.56868   3.56868 3.483904e-06
elb time_sec
1 0.9999983     0.39
```

We set `stop_rule = "equivalence"` to activate the stopping rule that is based on the equivalence theorem. In this case, ICA starts to calculate the ELB for the best design every `checkfreq` $= 20$ iterations and it stops whenever the value of the ELB is larger than `stoptol` $= 0.99$. In this example, ICA stopped at the first check run because the value of ELB is 0.999 ($>$ `stoptol`). Note that we requested to calculate the ELB after every 20 iterations, instead of every iteration, to prevent a significant increase in the CPU time. This equivalence-based stopping rule is also available in other OD functions. However, we note that optimality verification for Bayesian or minimax type criteria is more complicated and may slow down the ICA.

ICAOD can also handle a situation where the design points are pre-specified, but their optimal associated weights are of interest. For example, assume that the experimental resources only allow a pre-specified hours of practice, say, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ hours. In all OD functions, the design points can be specified similarly via the argument `x` (a vector of design points):

```
R> log3 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40,
+                  x = c(1, 2, 3),
+                  inipars = c(-4, 1.3333),
+                  ICA.control = list(rseed = 1, checkfreq = Inf))

R> print(log3)
Finding  locally optimal designs

Call:
  ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x))

iter x1 x2 x3        w1           w2        w3 min_cost mean_cost
1     1  1  2  3 0.4528099 2.460808e-03 0.5447293 4.196368  4.205840
5     5  1  2  3 0.5106454 5.660663e-03 0.4836939 4.190011  4.190011
9     9  1  2  3 0.4993132 8.104013e-05 0.5006058 4.187368  4.187368
```

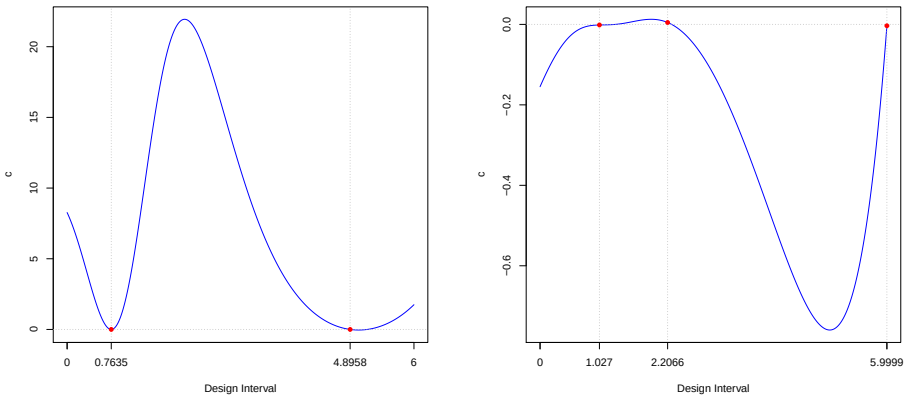
```
14   14  1  2  3 0.4993694 9.602963e-06 0.5006210 4.187346  4.187346
18   18  1  2  3 0.4998314 4.227502e-06 0.5001644 4.187343  4.187343
22   22  1  2  3 0.4998286 1.079043e-07 0.5001713 4.187342  4.187342
27   27  1  2  3 0.4999951 1.656952e-08 0.5000049 4.187342  4.187342
31   31  1  2  3 0.4999982 4.628899e-10 0.5000018 4.187342  4.187342
35   35  1  2  3 0.4999994 5.689118e-11 0.5000006 4.187342  4.187342
40   40  1  2  3 0.5000001 2.449702e-12 0.4999999 4.187342  4.187342

Optimal designs (k=3):
  Weights: 0.500 0.000 0.500

ICA iteration: 40
Criterion value:  4.187342
Total number of function evaluations: 1731
Total number of successful local search moves: 39
Total number of successful revolution moves: 30
Convergence: Maximum_Iteration
Total number of successful assimilation moves: 878
CPU time: 1.22  seconds!

  Maximum of the sensitivity function is  2.558775
Efficiency lower bound (ELB) is  0.4387143
Verification required 0.35 seconds!
```

The results show that no weight should be assigned to the subjects with 2 hours of practice. This means that, the responses from subjects with 2 hours of practice will not increase the efficiency of estimation very much. Hence, this level may be eliminated to save more resources.

The value of the ELB and the plot of the sensitivity function in Figure 1 (b) clearly show that the obtained design is not globally optimal. This comes as no surprise because the given design points in `x` do not belong to the support of the optimal design when $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. Note that `checkfreq = Inf` requests a `plot` method for the design provided by the output so that `plot()` is not required anymore. For space consideration, we use this option in the rest of this paper.

Locally optimal designs usually lose their efficiency when the parameter estimates are far from their true unknown values. Moreover, in practice, it is more realistic to assume that the parameters belong to a parameter space, rather than fixing their values at some points. For example, let $\boldsymbol{\theta} = (\beta_0, \beta_1)^T$ belongs to $\Theta = [\beta_0^L, \beta_0^U] \times [\beta_1^L, \beta_1^U]$, where $\beta_0^L = -6$, $\beta_0^U = -2$, $\beta_1^L = .5$ and $\beta_1^U = 2$. As a conservative strategy, a minimax $D-$optimal design minimizes the maximum inefficiency over Θ. To find the minimax $D-$optimal design for our design setting, we first set $k = 2$ to find the minimax $D-$optimal design within the class of two-point designs:

```
R> log4 <- minimax(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial",
+                  lx = 0, ux = 6, lp = c(-6, .5), up = c(-2, 2),
+                  iter = 200, k = 2,
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 50,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99),
+                  crt.minimax.control = list(optslist = list(maxeval = 200)))
R> print(log4)
Finding  minimax optimal designs

Call:
  ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x))

iter        x1       x2        w1        w2 min_cost mean_cost
1      1 0.4446832 4.868664 0.4243584 0.5756416 7.853582  8.061686
23    23 0.7665387 4.895727 0.4965758 0.5034242 7.782827  7.782827
45    45 0.7639494 4.895787 0.4999084 0.5000916 7.782754  7.782754
67    67 0.7636147 4.895791 0.5000079 0.4999921 7.782754  7.782754
89    89 0.7636144 4.895791 0.5000079 0.4999921 7.782754  7.782754
111  111 0.7636144 4.895791 0.5000079 0.4999921 7.782754  7.782754
133  133 0.7635408 4.895792 0.4999934 0.5000066 7.782754  7.782754
155  155 0.7635408 4.895792 0.4999934 0.5000066 7.782754  7.782754
```



(a) $k = 2$ **(b)** $k = 3$

Figure 2: Plots of the sensitivity functions of the two- and three-point designs generated by the `minimax()` function for the logistic regression model over $\chi = [0, 6]$ when $\Theta = [-6, -2] \times [0.5, 2]$. The left panel (a) does not verifies the optimality of the obtained design and the right panel (b) shows the nearly optimality of the three-point design. The solid red dots are the values of the sensitivity function at the obtained design points.

```
177  177 0.7635408 4.895792 0.4999934 0.5000066 7.782754  7.782754
200  200 0.7635408 4.895792 0.4999934 0.5000066 7.782754  7.782754

Optimal designs (k=2):
  Points1 Points2
0.76354 4.89579
Weights1 Weights2
0.500   0.500

ICA iteration: 200
Criterion value:  7.782754
Total number of function evaluations: 1710132
Total number of successful local search moves: 120
Total number of successful revolution moves: 60
Convergence: Maximum_Iteration
Total number of successful assimilation moves: 1007
Vector of maximum parameter values:  -6 0.5
CPU time: 211.47  seconds!

  Maximum of the sensitivity function is  21.9395
Efficiency lower bound (ELB) is  0.08354392
Verification required 0.72 seconds!
  Adjust the control parameters in 'sens.minimax.control' ('n_seg')
or in 'sens.bayes.control' for higher speed.
```

To increase the CPU time, we reduced the value of `maxeval` from 1000 (default value) to 200. Figure 2 (a) displays the sensitivity plot of the design by provided by the output and it does not verify the optimality of the two-point design. Therefore, we increment the value of k by one and re-execute the above code:

```
R> log5 <- minimax(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial",
+                  lx = 0, ux = 6, lp = c(-6, .5), up = c(-2, 2),
+                  iter = 500, k = 3,
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 50,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99),
+                  crt.minimax.control = list(optslist = list(maxeval = 200)))
```

```
R> print(log5)
Finding  minimax optimal designs

Call:
  ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x))

iter        x1       x2       x3        w1        w2        w3 min_cost
1     1 1.0613974 2.577126 5.994463 0.1263308 0.5337391 0.3399301 6.862938
6     6 0.9212932 2.161076 5.999569 0.1092450 0.5392481 0.3515069 6.826309
11   11 0.9291691 2.163344 5.998872 0.1131382 0.4602769 0.4265849 6.760707
17   17 0.9358663 2.165053 5.998143 0.1131382 0.4602769 0.4265849 6.758872
22   22 1.0343686 2.163586 5.997440 0.1000379 0.4592324 0.4407297 6.749718
28   28 1.1201234 2.340084 5.995084 0.1526103 0.3628279 0.4845619 6.745782
33   33 1.0084352 2.232590 5.999449 0.1185027 0.3770129 0.5044843 6.738361
39   39 1.0003813 2.245542 5.999894 0.1094120 0.3941922 0.4963958 6.737553
44   44 1.0159978 2.225392 5.999982 0.1148592 0.3916467 0.4934942 6.737084
50   50 1.0269974 2.206648 5.999927 0.1135038 0.3915064 0.4949898 6.736338
mean_cost
1   7.553929
6   6.951918
11  6.805511
17  6.776728
22  6.757709
28  6.748353
33  6.743650
39  6.742541
44  6.738447
50  6.737655

Optimal designs (k=3):
  Points1 Points2 Points3
1.02700 2.20665 5.99993
Weights1 Weights2 Weights3
0.114   0.392   0.495

ICA iteration: 50
Criterion value:  6.736338
Total number of function evaluations: 511836
Total number of successful local search moves: 309
Total number of successful revolution moves: 92
Convergence: equivalence
Total number of successful assimilation moves: 407
Vector of maximum parameter values:  -6 0.5
CPU time: 71.04  seconds!

  Maximum of the sensitivity function is  0.01269528
Efficiency lower bound (ELB) is  0.9936924
Verification required 2.16 seconds!
  Adjust the control parameters in 'sens.minimax.control' ('n_seg')
or in 'sens.bayes.control' for higher speed.
R> log5$design
iter       x1       x2       x3        w1        w2        w3 min_cost
1   50 1.026997 2.206648 5.999927 0.1135038 0.3915064 0.4949898 6.736338
mean_cost   max_sens       elb time_sec
1  6.737655 0.01269528 0.9936924     2.16
```

Figure 2 (b) displays the plot of the sensitivity function of the three-point generated design and it indicates its nearly optimality. The optimal design suggests subjects with nearly 1, 2 and 6 hours of practice, where roughly half of the subjects should be assigned to practice for 6 hours.

Similar to the locally $D-$optimal design, we can assess the minimax $D-$efficiency of ξ_{uni} with respect to the minimax $D-$optimal design by

```
R> meff(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+      predvars = "x", parvars = c("b0", "b1"),
+      family = "binomial",
```

```
+      lp = c(-6, .5), up = c(-2, 2),
+      x1 = c(0:6), w1 = rep(1/7, 7),
+      x2 = log5$evol[[20]]$x, w2 = log5$evol[[20]]$w)
[1] 0.7459795
```

This value indicates that ξ_{uni} requires about $100(1/0.74089 - 1) = 35\%$ more subjects to have the same minimax $D-$efficiency as the minimax $D-$optimal design when $\Theta = [-6, -2] \times [0.5, 2]$.

Sigmoid-Emax model

The sigmoid Emax model is commonly used in pharmacokinetics/pharamacodynamics to describe the S-shape dose-response relationship (see, e.g., Macdougall, 2006; Thomas, 2006). This model is defined by

$$E(Y) = f(x, \boldsymbol{\theta}) = \beta_1 + (\beta_2 - \beta_1)\frac{x^{\beta_4}}{x^{\beta_4} + \beta_3^{\beta_4}}, \tag{16}$$

where x is the dose level (in mg), $x \in \chi = (0, x_0]$, x_0 is user-selected and $\boldsymbol{\theta} = (\beta_1, \beta_2, \beta_3, \beta_4)^T$, $\theta_2 > \beta_1$, $\beta_3 > 0$. All errors are assumed to be independent and normally distributed with mean zero and constant variance. Here, β_1 is the minimum mean response, β_2 is the maximum mean response, β_3 is the ED50, i.e., the dose at which 50 percent of the maximum mean effect is achieved, and β_4 is the slope parameter.

In dose-response studies, optimal designs usually determine how many doses are required to be tested, what are their levels, and how many subjects to allocate to each dose level. Let $\chi = (0, 1000]$mg. Similar to Dragalin et al. (2007) and Wang and Yang (2014), we are interested in the efficient estimation of $\boldsymbol{\theta}$ and the $D-$optimality is an appropriate design criterion for this purpose.

It is straightforward to show that the FIM of the sigmoid Emax model depends on the unknown parameters $\boldsymbol{\theta}$. This parameter dependency must be dealt with based on the type of information available on $\boldsymbol{\theta}$. For example, using information from a pilot study, one may elicit a uniform prior distribution for $\boldsymbol{\theta}$ and search for Bayesian optimal designs. As an illustrative example, let $\beta_1 \sim U(4, 8)$, $\beta_2 \sim U(11, 15)$, $\beta_3 \sim U(100, 130)$ and $\beta_4 \sim U(5, 9)$, and all the uniform prior distributions be independent. For simplicity, we denote the independent uniform distributions for $\beta_i, i = 1, 2, 3, 4$ by π_Θ, where $\Theta = [4, 8] \times [11, 15] \times [100, 130] \times [5, 9]$ is the parameter space. This prior can be defined in **ICAOD** by the `uniform()` function as follows.

```
R> prior1 <- uniform(lower = c(4, 11, 100, 5), upper = c(8, 15, 130, 9))
```

Here, the output is an object of class '`cprior`', which can be passed to the argument `prior` of the `bayes()` function.

To find the number of support points for the Bayesian $D-$optimal design, we repeated the same incremental process for finding minimax optimal design. This process is excluded here due to space consideration. The Bayesian $D-$optimal design has 5 points in its support, which are found by

```
R> sig1 <- bayes(formula = ~b1 + (b2-b1) * x^b4/(x^b4 + b3^b4),
+                predvars = "x",
+                parvars = c("b1", "b2", "b3", "b4"),
+                lx = .001, ux = 1000, k = 5, iter = 400, prior = prior1,
+                ICA.control = list(rseed = 1, checkfreq = Inf))
R> print(sig1)
Finding  Bayesian optimal designs

Call:
  ~b1 + (b2 - b1) * x^b4/(x^b4 + b3^b4)

iter         x1       x2       x3       x4        x5        w1        w2
1      1 18.0475346 96.25014 167.5667 179.3485  867.8058 0.2909521 0.1581145
45    45 17.4961312 89.11527 108.3659 137.0731  866.5956 0.2368958 0.1548766
89    89  0.4786294 95.25557 115.0471 138.2732  554.3556 0.2460043 0.2036432
134  134  1.0415205 94.75054 113.8961 138.3159  959.5441 0.2430275 0.1959315
178  178  0.7994201 94.64836 113.7667 138.3264  999.8505 0.2433772 0.1949845
222  222  1.8666601 94.60760 113.7090 138.3516  999.9069 0.2432310 0.1942385
267  267  0.5492615 94.60194 113.7011 138.3539  999.9987 0.2432085 0.1941306
311  311  0.4263460 94.60176 113.6966 138.3513 1000.0000 0.2432061 0.1941301
355  355  0.4164132 94.60189 113.6965 138.3510 1000.0000 0.2432041 0.1941322
400  400  0.1805450 94.60188 113.6964 138.3510 1000.0000 0.2432040 0.1941319
```

```
w3          w4        w5 min_cost mean_cost
1   0.3418695 0.006883086 0.2021808 14.10396  15.82570
45  0.1428410 0.225597557 0.2397890 12.74113  12.74774
89  0.1043130 0.200569541 0.2454699 12.72302  12.72989
134 0.1140765 0.203490785 0.2434737 12.72086  12.72095
178 0.1150226 0.203128842 0.2434868 12.72082  12.72083
222 0.1158376 0.203138347 0.2435545 12.72082  12.72082
267 0.1159406 0.203152463 0.2435679 12.72082  12.72082
311 0.1159200 0.203173666 0.2435701 12.72082  12.72082
355 0.1159159 0.203177516 0.2435702 12.72082  12.72082
400 0.1159155 0.203178189 0.2435705 12.72082  12.72082

Optimal designs (k=5):
  Points1    Points2    Points3    Points4    Points5
0.18055    94.60188   113.69639  138.35096  1000.00000
Weights1   Weights2   Weights3   Weights4   Weights5
0.243      0.194      0.116      0.203      0.244

ICA iteration: 400
Criterion value:  12.72082
Total number of function evaluations: 85150
Total number of successful local search moves: 2378
Total number of successful revolution moves: 81
Convergence: maxiter
Total number of successful assimilation moves: 1700
CPU time: 611.44  seconds!

  Maximum of the sensitivity function is  9.439815e-07
Efficiency lower bound (ELB) is  0.9999998
Verification required 65.3 seconds!
  Adjust the control parameters in 'sens.minimax.control' ('n_seg')
or in 'sens.bayes.control' for higher speed.
R> sig1$design
iter       x1       x2       x3      x4   x5       w1        w2        w3
1  400 0.180545 94.60188 113.6964 138.351 1000 0.243204 0.1941319 0.1159155
w4        w5 min_cost mean_cost     max_sens       elb time_sec
1 0.2031782 0.2435705 12.72082  12.72082 9.439815e-07 0.9999998     65.3
```

Figure 3 (a) is generated from the output and presents the plot of the sensitivity function of the five-point design and it verifies its optimality. In our example, the Bayesian $D-$optimal design suggests five dose levels, with four of them located below 140mg and one located at the maximum. Roughly 50% of the observations should be assigned to the lower and upper bound of the dose interval. Note that the result can also be obtained in lesser CPU time if we adjust the control parameters of the integral approximations via the argument `crt.bayes.control`.

Using a non-optimal design may be inefficient even when its design points are sampled uniformly from the design space. As an illustrative example, assume a situation where a researcher decides to work with an equally-weighted uniform design that has 11 points located on $0.001, 100, 200, 300, ..., 1000$. This design is not optimal when $\boldsymbol{\theta} \sim \pi_\Theta$. The Bayesian $D-$efficiency of the uniform design with respect to the obtained Bayesian $D-$optimal design is calculated by

```
R> beff(formula = ~b1 + (b2-b1) * x ^b4/(x^b4 + b3^b4),
+      predvars = "x",
+      parvars = c("b1", "b2", "b3", "b4"),
+      prior = prior1,
+      x1 = c(.001,seq(100, 1000, by = 100)),
+      w1 = rep(1/11, 11),
+      x2 = sig1$evol[[400]]$x, w2 = sig1$evol[[400]]$w)
[1] 0.3063289
```

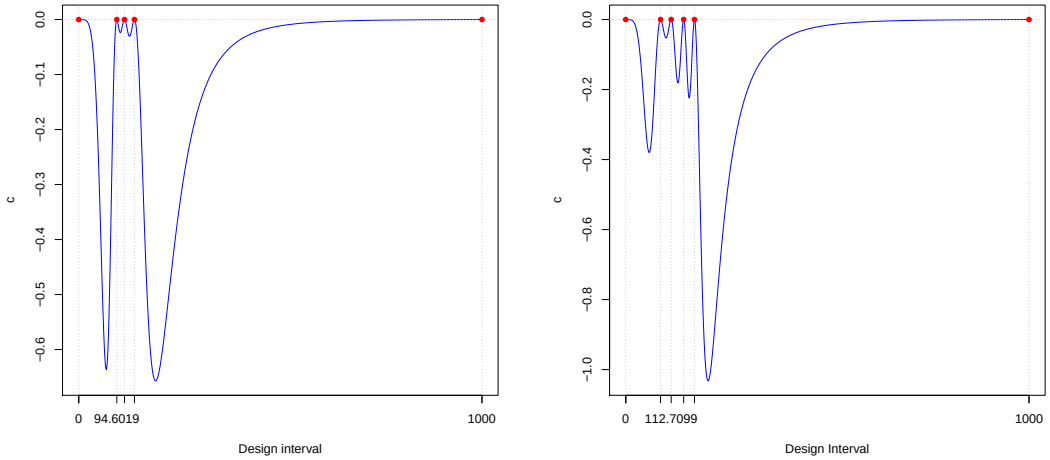
The non-optimal design may seem reasonable, but its Bayesian $D-$efficiency value suggests that, roughly 226% more observations are needed to maintain the $D-$efficiency for the non-optimal design in comparison to the Bayesian $D-$optimal design when $\boldsymbol{\theta} \sim \pi_\Theta$. The `bayes()` function is very flexible and can incorporate different prior distributions.

ICAOD can also find robust or optimum-on-average designs when the prior distributions are discrete. As an illustrative example, assume $\Theta_0 = \{\boldsymbol{\theta}_{01}, \boldsymbol{\theta}_{02}, \boldsymbol{\theta}_{03}, \boldsymbol{\theta}_{04}, \boldsymbol{\theta}_{05}\}$ be a set of five vectors

of initial estimates for $\boldsymbol{\theta} = (\beta_1, \beta_2, \beta_3, \beta_4)$, where $\boldsymbol{\theta}_{01} = (4, 11, 100, 5)$, $\boldsymbol{\theta}_{02} = (5, 12, 110, 6)$, $\boldsymbol{\theta}_{03} = (6, 13, 120, 7)$, $\boldsymbol{\theta}_{04} = (8, 15, 130, 9)$ and $\boldsymbol{\theta}_{05} = (12, 30, 160, 13)$. Let π_{Θ_0} denotes a discrete uniform prior distribution that assigns the same probability to each vector element of Θ_0. The six-point optimum-on-average design is given by

```
R> parset1 <- matrix(c(4, 11, 100, 5,
+                      5, 12, 110, 6,
+                      6, 13, 120, 7,
+                      8, 15, 130, 9,
+                      12, 30, 160, 13),
+                    nrow = 5, byrow = TRUE)
R> sig2 <- robust(formula = ~b1 + (b2-b1) * x ^b4/(x^b4 + b3^b4),
+                 predvars = "x",
+                 parvars = c("b1", "b2", "b3", "b4"),
+                 lx = .001, ux = 1000, k = 6, iter = 400,
+                 parset = parset1,
+                 prob = rep(1/5, 5),
+                 ICA.control = list(rseed = 1, checkfreq = Inf))
R> print(sig2)
Finding  robust or optimum-on-average optimal designs

Call:
  ~b1 + (b2 - b1) * x^b4/(x^b4 + b3^b4)

iter          x1       x2       x3       x4       x5        x6        w1
1      1 52.47474098 86.13143 108.6089 176.5576 847.1196  865.1056 0.3356308
45    45  0.03699118 93.84970 115.0752 144.3279 172.2496  612.5133 0.1921861
89    89  0.26057011 86.13743 112.6560 143.7663 170.6661  899.7495 0.1938690
134  134  0.27440675 86.41506 112.7178 143.7321 170.5697  999.4115 0.1997277
178  178  0.42978429 86.41957 112.7179 143.7262 170.5713  999.5817 0.2001652
222  222  0.86217693 86.41953 112.7092 143.7262 170.5733  999.9999 0.2001538
267  267  0.78134061 86.42156 112.7098 143.7250 170.5722 1000.0000 0.2001708
311  311  0.28372653 86.42155 112.7098 143.7248 170.5723 1000.0000 0.2001738
355  355  0.08123600 86.42156 112.7099 143.7248 170.5723 1000.0000 0.2001734
400  400  0.04980091 86.42158 112.7099 143.7248 170.5723 1000.0000 0.2001734
w2        w3        w4         w5        w6 min_cost mean_cost
1   0.06870259 0.1056298 0.2022530 0.12858302 0.1592008 14.10402  16.35115
45  0.15186984 0.1444531 0.2155502 0.08196997 0.2139708 12.25422  12.31711
89  0.13222946 0.1570768 0.1882862 0.09776625 0.2307723 12.21447  12.28391
134 0.13190606 0.1549309 0.1855214 0.09816873 0.2297452 12.21398  12.28328
178 0.13154222 0.1547794 0.1858112 0.09840028 0.2293017 12.21398  12.28311
222 0.13149988 0.1548130 0.1857954 0.09845378 0.2292841 12.21398  12.28305
267 0.13150916 0.1547914 0.1857821 0.09847028 0.2292762 12.21398  12.27535
311 0.13150554 0.1547882 0.1857820 0.09847434 0.2292761 12.21398  12.26001
355 0.13150671 0.1547883 0.1857817 0.09847396 0.2292760 12.21398  12.21398
400 0.13150681 0.1547882 0.1857817 0.09847394 0.2292759 12.21398  12.21398

Optimal designs (k=6):
  Points1    Points2    Points3    Points4    Points5    Points6
0.04980    86.42158   112.70988  143.72485  170.57227  1000.00000
Weights1   Weights2   Weights3   Weights4   Weights5   Weights6
0.200      0.132      0.155      0.186      0.098      0.229

ICA iteration: 400
Criterion value:  12.21398
Total number of function evaluations: 20070
Total number of successful local search moves: 3787
Total number of successful revolution moves: 88
Convergence: Maximum_Iteration
Total number of successful assimilation moves: 1895
CPU time: 29.56  seconds!

  Maximum of the sensitivity function is  3.960066e-07
Efficiency lower bound (ELB) is  0.9999999
Verification required 2.27 seconds!
```

```
  R> sig2$design
iter          x1       x2       x3       x4       x5   x6        w1        w2
1  400 0.04980091 86.42158 112.7099 143.7248 170.5723 1000 0.2001734 0.1315068
w3        w4         w5        w6 min_cost mean_cost     max_sens
1 0.1547882 0.1857817 0.09847394 0.2292759 12.21398  12.21398 3.960066e-07
elb time_sec
1 0.9999999     2.27
```



(a) $\boldsymbol{\theta} \sim \pi_{\Theta}$ **(b)** $\boldsymbol{\theta} \sim \pi_{\Theta_0}$

Figure 3: The plots of sensitivity functions of the generated designs for the sigmoid Emax model over the design space $[0.001, 1000]$. The left panel (a) displays the plot of the sensitivity function of the design generated by the function `bayes()` when $\boldsymbol{\theta} \sim \pi_{\Theta}$. The right panel (b) displays the plot of the sensitivity function of the design generated by the function `robust()` when $\boldsymbol{\theta} \sim \pi_{\Theta_0}$. Both (a) and (b) verify the optimality of the obtained design. The solid red dots are the values of the sensitivity function at the obtained design points.

Figure 3 (b) displays the plot of the sensitivity function of the design provided by the output and it verifies the optimality of the six-point design. Similar to the optimal design generated by `bayes()`, the generated design here allocates most of its support points to the lower half of the dose interval.

6 User-specified optimality criteria

ICAOD can also find optimal designs with respect to user-specified optimality criteria. In this section, as an illustrative example, we find $c-$optimal designs for the two-parameter logistic (2PL) model with applications in dose-response studies. The 2PL model is commonly used in dose-response studies to model the relationship between the dose level of a drug and the probability of a success, e.g., the probability that patients are cured. This model is defined by

$$f(x, \boldsymbol{\theta}) = P(Y = 1) = \frac{1}{1 + \exp(-b(x - a))}, \tag{17}$$

where x is the dose level (predictor), $\boldsymbol{\theta} = (a, b)^T$, b is the slope parameter and a is the dose level at which the response probability is 0.5 (ED50). Throughout this paper, we denote the dose level at which the response probability is equal to π by ED100π. For the 2PL model, it can be shown that ED100π is equal to $c(\boldsymbol{\theta}) = a + \gamma b^{-1}$, where $\gamma = \log[\pi/(1-\pi)]$ (see, e.g., Zhu and Wong, 2001).

Sometimes the purpose of a study is to estimate a function of the unknown parameters, say, ED100π, rather than estimating all the parameters simultaneously. For example, in heart defibrillator design problems, estimating the ED95, or equivalently, estimating $c(\boldsymbol{\theta}) = a + \log(0.95/(1-0.95))b^{-1}$ for the 2PL model is of interest (Clyde et al., 1995). In this case, a reasonable optimality criterion is the one that minimizes the asymptotic variance of the maximum likelihood (ML) estimator of $c(\boldsymbol{\theta})$, which is proportional to

$$\psi^c(\xi, \boldsymbol{\theta}) = \nabla^T c(\boldsymbol{\theta}) M^{-1}(\xi, \boldsymbol{\theta}) \nabla c(\boldsymbol{\theta}), \tag{18}$$

where $\nabla c(\boldsymbol{\theta})$ is the gradient of $c(\boldsymbol{\theta})$ and $M^{-1}(\xi, \boldsymbol{\theta})$ is the inverse of the FIM (see, e.g., Silvey, 1980, page 4). For the 2PL model, $\nabla c(\boldsymbol{\theta}) = (1, -\gamma b^{-2})^T$. In the optimal design literature, $\psi^c(\xi, \boldsymbol{\theta})$ is referred to as $c-$optimality criterion and a design that minimizes $\psi^c(\xi, \boldsymbol{\theta})$ is called $c-$optimal design. An equivalence theorem is also available for $c-$optimality: a design ξ_c^* is $c-$optimal among all the designs on χ if and only if the following inequality holds for all $x \in \chi$,

$$c^c(x, \xi_c^*) = \text{tr}(B(\boldsymbol{\theta})M^{-1}(\xi, \boldsymbol{\theta})M(\xi_x, \boldsymbol{\theta})M^{-1}(\xi, \boldsymbol{\theta})) - \psi^c(\xi, \boldsymbol{\theta}) \leq 0, \tag{19}$$

with equality in (19) for all the support points of ξ_c^* (see, e.g., Chaloner and Larntz, 1989). Here, $B(\boldsymbol{\theta}) = \nabla^T c(\boldsymbol{\theta}) \nabla c(\boldsymbol{\theta})$ and ξ_x denotes a degenerate design that puts all its mass on x.

Similar to the $D-$optimality criterion, $c-$optimality also depends on the unknown parameters and different types of optimal designs may be found, depending on how to deal with the unknown parameters. As benchmark examples, in this section, we find locally and Bayesian $c-$optimal designs for estimating the ED95 for the 2PL model when $\chi = [-1, 1]$. These examples are also available in Chaloner and Larntz (1989). Finding a minimax $c-$optimal or a robust design is very similar and is excluded due to space consideration.

To use **ICAOD** for finding $c-$optimal designs, the user should first define the $c-$optimality criterion and its sensitivity function as two separate functions in the R environment. Later, these functions will be passed to `bayes()`, `minimax()`, `locally()` and `robust()` via the `crtfunc` and `sensfunc` arguments, respectively. For example, given the 2PL model with parameters `parvars = c("a","b")`, the following lines of codes define (18) and (19) in the R environment to be used in `locally()`, `minimax()` and `robust()`.

```
R> c_opt <-function(x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   c <- matrix(c(1, -gam * b^(-2)), nrow = 1)
+   B <- t(c) %*% c
+   sum(diag(B %*% solve(M)))
+ }

R> c_sens <- function(xi_x, x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   M_inv <- solve(M)
+   M_x <- fimfunc(x = xi_x, w = 1, a = a, b = b)
+   c <- matrix(c(1, -gam * b^(-2)), nrow = 1)
+   B <- t(c) %*% c
+   sum(diag(B %*% M_inv %*% M_x %*%  M_inv)) - sum(diag(B %*% M_inv))
+ }
```

The arguments `x`, `w` are, respectively, the vector of design points and their associated weights defined by (2). `fimfunc()` is a function with arguments `x`, `w`, `a` and `b` that returns the evaluated FIM as a `matrix` and `xi_x` denotes a degenerate design, which has the same length as the number of model predictors. The arguments `a` and `b` are model-specific and denote the parameters of the model that is specified via `parvars`. A convenient feature of **ICAOD** is that there is no need to compute the FIM of the model even for a user-specified optimality criterion and the user can apply the internally-created FIM within the body of `c_opt()` and `c_sens()` using `fimfunc()`. Note that both of the `c_opt()` and `c_sens()` functions are not vectorized with respect to `a` and `b`. This means that `fimfunc()` returns only a `matrix`, and `c_opt()` and `c_sens()` return a value. This is a necessary structure required by the `locally()`, `minimax()` and `robust()` functions. The following lines of codes provide the locally $c-$optimal design for estimating the ED95 when $\boldsymbol{\theta} = (0, 7)$.

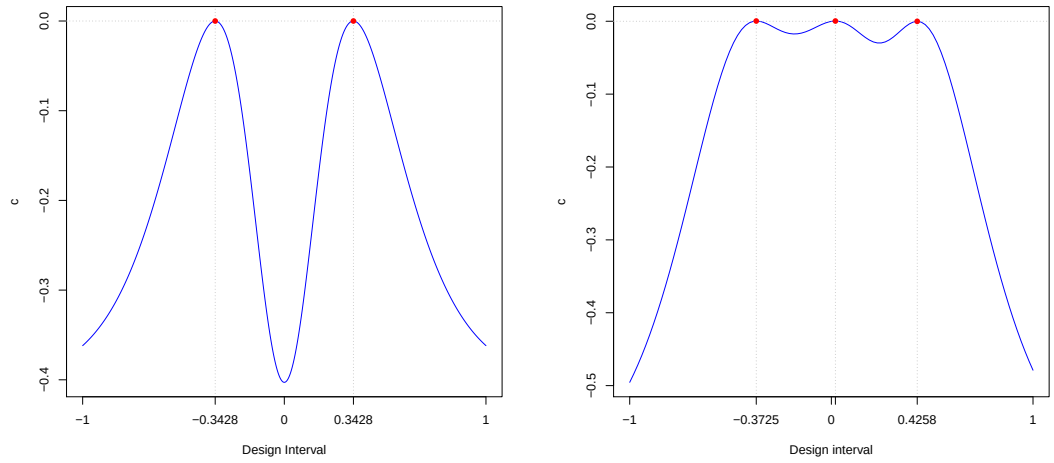
```
R> twoPL1 <- locally(formula = ~1/(1 + exp(-b * (x-a))), predvars = "x",
+                    parvars = c("a", "b"), family = "binomial",
+                    lx = -1, ux = 1, inipars = c(0, 7),
+                    iter = 100, k = 2,
+                    crtfunc = c_opt, sensfunc = c_sens,
+                    ICA.control = list(rseed = 1, checkfreq = Inf))
R> print(twoPL1)
Finding  locally optimal designs

Call:
  ~1/(1 + exp(-b * (x - a)))
```

```
iter         x1        x2         w1        w2  min_cost mean_cost
1      1 -0.5925888 0.3010654 0.18893987 0.8110601 0.4610712 0.5538651
12    12 -0.3542959 0.3287717 0.11649268 0.8835073 0.4038230 0.4152469
23    23 -0.3319221 0.3409457 0.09387297 0.9061270 0.4028975 0.4061873
34    34 -0.3427433 0.3430464 0.09225956 0.9077404 0.4028270 0.4030954
45    45 -0.3427747 0.3427851 0.09254018 0.9074598 0.4028266 0.4028356
56    56 -0.3427642 0.3427662 0.09255921 0.9074408 0.4028266 0.4028277
67    67 -0.3427648 0.3427655 0.09256089 0.9074391 0.4028266 0.4028271
78    78 -0.3427653 0.3427653 0.09256118 0.9074388 0.4028266 0.4028268
89    89 -0.3427653 0.3427653 0.09256120 0.9074388 0.4028266 0.4028267
100  100 -0.3427653 0.3427653 0.09256119 0.9074388 0.4028266 0.4028266

Optimal designs (k=2):
  Points1  Points2
-0.34277 0.34277
Weights1 Weights2
0.093    0.907

ICA iteration: 100
Criterion value:  0.4028266
Total number of function evaluations: 4764
Total number of successful local search moves: 415
Total number of successful revolution moves: 65
Convergence: Maximum_Iteration
Total number of successful assimilation moves: 1157
CPU time: 4.09  seconds!

  Maximum of the sensitivity function is  1.181344e-09
Efficiency lower bound (ELB) is  1
Verification required 0.69 seconds!
  R> twoPL1$design
iter         x1        x2         w1        w2  min_cost mean_cost
1  100 -0.3427653 0.3427653 0.09256119 0.9074388 0.4028266 0.4028266
max_sens elb time_sec
1 1.181344e-09   1     0.69
```

The obtained design suggests that nearly 90% of the observations should be assigned to 0.34277 and the rest should be allocated to -0.34277. Figure 4 (a) displays the plot of the sensitivity function of the obtained design and it indicates its optimality. Using the given `c_opt()` and `c_sens()` functions, we can similarly find minimax $c-$optimal or robust designs. For illustrating example, see `?minimax` and `?robust`.

Finding Bayesian $c-$optimal design is very similar, except that each of (18) and (19) must be a vectorized R `function` with respect to the model parameters `a` and `b`:

```
R> c_opt_vec <-function(x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   B <- sapply(1:length(M), FUN = function(i)
+     matrix(c(1, -gam * b[i]^(-2)), ncol= 1) %*%
+       matrix(c(1, -gam * b[i]^(-2)), nrow = 1), simplify = FALSE)
+   sapply(1:length(M), FUN = function(i)
+     sum(diag(B[[i]] %*% solve(M[[i]]))))
+ }
R> c_sens_vec <- function(xi_x, x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95)) # LD .95
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   M_inv <- lapply(M , FUN = function(FIM) solve(FIM))
+   M_x <- fimfunc(x = xi_x, w = 1, a = a, b = b)
+   B <- sapply(1:length(M), FUN = function(i)
+     matrix(c(1, -gam * b[i]^(-2)), ncol= 1) %*%
+       matrix(c(1, -gam * b[i]^(-2)), nrow = 1), simplify = FALSE)
+   sapply(1:length(M), FUN = function(i)
+     sum(diag(B[[i]] %*% M_inv[[i]] %*% M_x[[i]] %*% M_inv[[i]])) -
+       sum(diag(B[[i]] %*% M_inv[[i]])))
+ }
```



(a) `Output from plot(twoPL1)` **(b)** `Output from plot(twoPL2)`

Figure 4: Plots of the sensitivity functions of the generated $c-$optimal designs for estimating the ED95 when $x \in \chi = [-1, 1]$. The left panel (a) displays the sensitivity function of the locally $c-$optimal design when $\boldsymbol{\theta} = (0, 7)$. The right panel (b) displays the sensitivity function of the Bayesian $c-$optimal design when $a \sim U(-0.3, 0.3)$ and $b \sim U(6, 8)$. Both (a) and (b) verify the optimality of the obtained designs. The solid red dots are the values of the sensitivity function at the obtained design points.

In the `c_opt_vec` and `c_sens_vec` functions, the arguments `a` and `b` are now vectors of the same (dynamic) length, and `fimfunc()` now returns a `list` of matrices with length equal to `length(a)`. Let $a \sim U(-0.3, 0.3)$ and $b \sim U(6, 8)$. Given `c_opt_vec` and `c_sens_vec`, the Bayesian $c-$optimal design for estimating the ED95 is obtained by

```
R> twoPL2 <- bayes(formula = ~1/(1 + exp(-b * (x-a))), predvars = "x",
+                  parvars = c("a", "b"), family = "binomial",
+                  lx = -1, ux = 1,
+                  prior = uniform(lower = c(-.3, 6), upper  = c(.3, 8)),
+                  iter = 100, k = 3,
+                  crtfunc = c_opt_vec,
+                  sensfunc = c_sens_vec,
+                  ICA.control = list(rseed = 1, ncount = 60, checkfreq = Inf),
+                  sens.bayes.control = list(cubature = list(maxEval = 100)))
R> print(twoPL2)
Finding  Bayesian optimal designs

Call:
  ~1/(1 + exp(-b * (x - a)))

iter           x1         x2        x3         w1        w2        w3
1      1 -0.004009039 0.29828119 0.4605856 0.19080601 0.3821834 0.4270106
12    12  0.004988270 0.40974945 0.4383355 0.20244712 0.2165999 0.5809529
23    23 -0.353135960 0.03376940 0.4250446 0.04238018 0.2424870 0.7151328
34    34 -0.351030679 0.03608485 0.4284948 0.04107568 0.2404412 0.7184831
45    45 -0.370795161 0.02795782 0.4282277 0.03956796 0.2269339 0.7334982
56    56 -0.326444221 0.02699155 0.4266908 0.02763386 0.2197316 0.7526345
67    67 -0.359979100 0.02467998 0.4271671 0.02883547 0.2165489 0.7546157
78    78 -0.374296800 0.02160651 0.4265674 0.02691817 0.2180796 0.7550023
89    89 -0.372327659 0.01982217 0.4257716 0.02629106 0.2186265 0.7550825
100  100 -0.372518847 0.02002258 0.4257626 0.02640997 0.2186892 0.7549009
min_cost mean_cost
1   0.6555717 0.7701302
12  0.6288037 0.6371994
23  0.6267816 0.6297125
34  0.6265897 0.6281724
```

```
45  0.6258766 0.6279520
56  0.6254187 0.6278212
67  0.6253270 0.6277844
78  0.6252741 0.6277719
89  0.6252610 0.6277691
100 0.6252608 0.6277691

Optimal designs (k=3):
  Points1  Points2  Points3
-0.37252 0.02002  0.42576
Weights1 Weights2 Weights3
0.026    0.219    0.755

ICA iteration: 100
Criterion value:  0.6252608
Total number of function evaluations: 38461
Total number of successful local search moves: 1087
Total number of successful revolution moves: 134
Convergence: maxiter
Total number of successful assimilation moves: 1115
CPU time: 203.82  seconds!

  Maximum of the sensitivity function is  0.0003369562
Efficiency lower bound (ELB) is  0.9998316
Verification required 3.56 seconds!
  Adjust the control parameters in 'sens.minimax.control' ('n_seg')
or in 'sens.bayes.control' for higher speed.
R> twoPL2$design
iter         x1         x2        x3         w1        w2        w3  min_cost
1  100 -0.3725188 0.02002258 0.4257626 0.02640997 0.2186892 0.7549009 0.6252608
mean_cost     max_sens       elb time_sec
1 0.6277691 0.0003369562 0.9998316     3.56
```

Figure 4 (b) displays the plot of the sensitivity function of the design provided by the output and it verifies its optimality. Similar to the locally $c-$optimal design, this design puts more than 97% of its weight on the positive support points.

7 Summary

ICAOD modifies a state-of-the-art metaheuristic algorithm called Imperialist Competitive Algorithm to find different types of optimal designs for nonlinear models. We believe this package is more self-contained and has more capability than the few available in the literature. In particular, **ICAOD** offers different design approaches for handling the parameter dependency in the information matrix when the model is nonlinear. A useful feature of the **ICAOD** package is that it can create the Fisher information matrices for a very general class of nonlinear models automatically and also includes useful theory-based tools to assess proximity of any design to the optimal design without knowing the latter. Using **ICAOD**, it is also possible to find optimal designs for a user-specified optimality criterion, including hard-to-find various types of minimax optimal designs for which the criterion is not differentiable.

Due to space consideration, we presented only a few examples in this paper to show the functionality of the package. The help-documentation manual for the package contains further details and illustrations. We hope that the generality and simplicity of the **ICAOD** package will encourage researchers from different disciplines to explore optimal design ideas in their work and enable them to implement a more informed design to realize maximum statistical efficiency at minimal cost.

Computational details

The results in this paper were obtained using R 4.0.2 with the **ICAOD** 1.0.1 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/.

Acknowledgments

We would like to thank Dr. Paul-Christian Bürkner for his helpful comments when writing the package and this manuscript. The research of Wong reported in this paper was partially supported by a grant award R01GM107639 from the National Institute of General Medical Sciences of the National Institutes of Health. The research of Holling and Masoudi in this paper was partially supported by a grant (HO 1286/6 - 4) of the German Research Foundation (DFG). The research of Kim reported in this paper was partially supported by grant awards R21GM140352 from the National Institute of General Medical Sciences and P30CA022453 from the National Cancer Institute of the National Institutes of Health. The contents in this paper are solely the responsibility of the authors and do not necessarily represent the official views of the National Institutes of Health.

Disclosure

Ehsan Masoudi is an employee of Roche Pharma AG in Germany since October 2019.

Bibliography

K. M. Abdelbasit and R. L. Plackett. Experimental design for binary data. *Journal of the American Statistical Association*, 78(381):90–98, 1983. ISSN 01621459. doi: 10.1080/01621459.1983.10477936. [p20]

E. Atashpaz-Gargari and C. Lucas. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *IEEE congress on evolutionary computation*, pages 4661–4667, 2007. doi: 10.1109/CEC.2007.4425083. [p21, 23, 24]

A. C. Atkinson. The usefulness of optimum experimental designs. *Journal of the Royal Statistical Society B*, 58:59–76, 1996. doi: 10.1111/j.2517-6161.1996.tb02067.x. [p20]

C. L. Atwood. Optimal and efficient designs of experiments. *The Annals of Mathematical Statistics*, 40(5):1570–1602, 1969. doi: 10.1214/aoms/1177697374. [p23]

M. P. Berger and W. K. Wong. *Applied optimal designs*. Jonh Wiley & Sons, Chichester, West Sussex, UK, 2005. doi: 10.1002/0470857005. [p27]

M. P. Berger and W. K. Wong. *An introduction to optimal designs for social and biomedical research*. Jonh Wiley & Sons, Chichester, West Sussex, UK, 2009. doi: 10.1002/9780470746912. [p20]

M. P. Berger, C. J. King, and W. K. Wong. Minimax D-optimal designs for item response theory models. *Psychometrika*, 65(3):377–390, 2000. doi: 10.1007/BF02296152. [p23]

J. Berntsen, T. O. Espelid, and A. Genz. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions on Mathematical Software (TOMS)*, 17(4):437–451, 1991. doi: 10.1145/210232.210233. [p27]

P.-C. Bürkner, R. Schwabe, and H. Holling. Optimal designs for the generalized partial credit model. *British Journal of Mathematical and Statistical Psychology*, 2019. doi: 10.1111/bmsp.12148. [p20]

K. Chaloner and K. Larntz. Optimal Bayesian design applied to logistic regression experiments. *Journal of Statistical Planning and Inference*, 21(2):191–208, 1989. doi: 10.1016/0378-3758(89)90004-9. [p20, 23, 38]

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995. doi: 10.1214/ss/1177009939. [p20]

H. Chernoff. Locally optimal designs for estimating parameters. *The Annals of Mathematical Statistics*, 24(4):586–602, 1953. doi: 10.1214/aoms/1177728915. [p20]

M. Clyde, P. Müller, and G. Parmigiani. Optimal design for heart defibrillators. In *Case Studies in Bayesian Statistics, Volume II*, pages 278–292. Springer, 1995. doi: 10.1007/978-1-4612-2546-1_7. [p37]

H. Dette, C. Kiss, M. Bevanda, and F. Bretz. Optimal designs for the EMAX, log-linear and exponential models. *Biometrika*, 97(2):513–518, 2010. doi: 10.1093/biomet/asq020. [p20]

V. Dragalin, F. Hsuan, and S. K. Padmanabhan. Adaptive designs for dose-finding studies based on sigmoid e max model. *Journal of Biopharmaceutical Statistics*, 17(6):1051–1070, 2007. doi: 10.1080/10543400701643954. [p34]

V. Fedorov. Convex design theory. *Series Statistics*, 11(3):403–413, 1980. doi: 10.1080/02331888008801549. [p23]

V. V. Fedorov and P. Hackl. *Model-Oriented Design of Experiments*, volume 125. Springer Science & Business Media, 2012. doi: 10.1007/978-1-4612-0703-0. [p20]

V. V. Fedorov and S. L. Leonov. *Optimal design for nonlinear response models*. CRC Press, 2013. [p20, 22]

D. Firth and J. Hinde. On bayesian d-optimum design criteria and the equivalence theorem in non-linear models. *Journal of the Royal Statistical Society B*, 59(4):793–797, 1997. doi: 10.1111/1467-9868.00096. [p20]

J. M. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21(1):27–37, 2001. doi: 10.1023/A:1017930332101. [p25]

A. Genz and A. Malik. Remarks on algorithm 006: An adaptive algorithm for numerical integration over an n-dimensional rectangular region. *Journal of Computational and Applied Mathematics*, 6 (4):295 – 302, 1980. ISSN 0377-0427. doi: 10.1016/0771-050X(80)90039-X. [p27]

U. Graßhoff, H. Holling, and R. Schwabe. Optimal designs for the rasch model. *Psychometrika*, 77 (4):710–723, 2012. doi: 10.1007/s11336-012-9276-2. [p20]

H. Holling and R. Schwabe. An introduction to optimal design: Some basic issues using examples from dyscalculia research. *Zeitschrift für Psychologie*, 221(3):124, 2013. [p20]

S. Hosseini and A. Al Khaled. A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. *Applied Soft Computing*, 24:1078–1094, 2014. doi: 10.1016/j.asoc.2014.08.024. [p21, 24]

S. W. Hyun, W. K. Wong, and Y. Yang. VNM: An R package for finding multiple-objective optimal designs for the 4-parameter logistic model. *Journal of Statistical Software*, 83(5):1–19, 2018. doi: 10.18637/jss.v083.i05. [p21]

S. G. Johnson. Cubature package, 2013. URL https://github.com/stevengj/cubature. Version 1.0.3. [p26]

S. G. Johnson. The NLopt nonlinear-optimization package, 2014. URL http://ab-initio.mit.edu/nlopt. [p25]

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968. [p21]

J. Kiefer and J. Wolfowitz. Optimum designs in regression problems. *The Annals of Mathematical Statistics*, 30:271–294, 1959. doi: 10.1214/aoms/1177706252. [p23]

J. King and W. K. Wong. Minimax D-optimal designs for the logistic model. *Biometrics*, 56(4): 1263–1267, 2000. [p23]

E. L. Lehmann and G. Casella. *Theory of point estimation*, volume 31. Springer Science & Business Media, 1998. [p20]

J. L. Lin, C. W. Cho, and H. C. Chuan. Imperialist competitive algorithms with perturbed moves for global optimization. In *Applied Mechanics and Materials*, volume 284, pages 3135–3139. Trans Tech Publ, 2013. doi: 10.4028/www.scientific.net/amm.284-287.3135. [p24]

J. Macdougall. Analysis of dose–response studies–e max model. In *Dose finding in drug development*, pages 127–145. Springer, New York, NY, 2006. doi: 10.1007/0-387-33706-7_9. [p34]

E. Masoudi, M. Sarmad, and H. Talebi. *LDOD: Finding Locally D-optimal Optimal Designs for Some Nonlinear and Generalized Linear Models.*, 2013. URL http://CRAN.R-project.org/package=LDOD. R package version 1.0. [p21]

E. Masoudi, H. Holling, and W. K. Wong. Application of imperialist competitive algorithm to find minimax and standardized maximin optimal designs. *Computational Statistics & Data Analysis*, 113:330–345, 2017. doi: 10.1016/j.csda.2016.06.014. [p21, 23]

E. Masoudi, H. Holling, B. P. Duarte, and W. K. Wong. Metaheuristic adaptive cubature based algorithm to find bayesian optimal designs for nonlinear models. *Journal of Computational and Graphical Statistics*, 2019. doi: 10.1080/10618600.2019.1601097. [p21]

E. Masoudi, H. Holling, W. K. Wong, and S. Kim. ICAOD: Optimal designs for nonlinear models, 2020. URL http://CRAN.R-project.org/package=ICAOD. R package version 1.0.0. [p21]

B. Narasimhan and S. G. Johnson. cubature: Adaptive multivariate integration over hypercubes, 2017. URL https://CRAN.R-project.org/package=cubature. R package version 1.3-11. [p26]

A. Overstall, D. Woods, and M. Adamou. acebayes: An r package for bayesian optimal design of experiments via approximate coordinate exchange. 2017. URL arXivpreprintarXiv:1705.08096. [p21]

A. M. Overstall and D. C. Woods. Bayesian design of experiments using approximate coordinate exchange. *Technometrics*, 59(4):458–470, 2017. doi: 10.1080/00401706.2016.1251495. [p21]

A. Pázman. *Foundations of optimum experimental design*, volume 14. Springer, 1986. [p23]

F. Pukelsheim and S. Rieder. Efficient rounding of approximate designs. *Biometrika*, 79(4):763–770, 1992. doi: 10.2307/2337232. [p22]

SAS Institute Inc. *JMP® 13 Design of Experiments Guide*. Cary, NC: SAS Institute Inc, 2016. URL https://www.jmp.com. [p21]

S. D. Silvey. *Optimal design : an introduction to the theory for parameter estimation*. London; New York : Chapman and Hall, 1980. doi: 10.1007/978-94-009-5912-5. [p22, 38]

R. R. Sitter. Robust designs for binary data. *Biometrics*, 48(4):1145–1155, 1992. doi: 10.2307/2532705. [p20]

N. Thomas. Hypothesis testing and bayesian estimation using a sigmoid e max model applied to sparse dose-response designs. *Journal of Biopharmaceutical Statistics*, 16(5):657–677, 2006. doi: 10.1080/10543400600860469. [p34]

T. Wang and M. Yang. Adaptive optimal designs for dose-finding studies based on sigmoid emax models. *Journal of Statistical Planning and Inference*, 144:188–197, 2014. doi: 10.1016/j.jspi.2013.09.003. [p34]

C. Weiser. mvQuad: Methods for multivariate quadrature, 2016. URL http://CRAN.R-project.org/package=mvQuad. R package version 1.0-6. [p27]

W. K. Wong. A unified approach to the construction of minimax designs. *Biometrika*, 79(3):611–619, 1992. doi: doi.org/10.1093/biomet/79.3.611. [p23]

X.-S. Yang. Metaheuristic optimization: Algorithm analysis and open problems. In P. M. Pardalos and S. Rebennack, editors, *Experimental Algorithms*, pages 21–32, Berlin, 2011. Springer-Verlag. doi: 10.1007/978-3-642-20662-7_2. [p21]

W. Zhu and W. K. Wong. Bayesian optimal designs for estimating a set of symmetrical quantiles. *Statistics in Medicine*, 20(1):123–137, 2001. doi: 10.1002/1097-0258(20010115)20:1<123::aid-sim643>3.0.co;2-5. [p37]

Ehsan Masoudi
Department of Psychology, University of Münster
Fliednerstr. 21, 48149 Germany
esn_mud@yahoo.com

Heinz Holling
Department of Psychology, University of Münster
Fliednerstr. 21, 48149 Germany
holling@uni-muenster.de

Weng Kee Wong
Department of Biostatistics
UCLA Fielding School of Public Health

Los Angeles, CA 90095-1772, USA
wkwong@ucla.edu

Seongho Kim
Biostatistics and Bioinformatics Core, Karmanos Cancer Institute
Department of Oncology, Wayne State University School of Medicine
Detroit, MI 48201, USA
kimse@karmanos.org

Analysis of the Results of Metadynamics Simulations by metadynminer and metadynminer3d

by Dalibor Trapl and Vojtech Spiwok

Abstract Molecular simulations solve the equation of motion of molecular systems, making the 3D shapes of molecules four-dimensional by adding the time coordinate. These methods have great potential in drug discovery because they can realistically model the structures of protein molecules targeted by drugs, as well as the process of binding of a potential drug to its molecular target. However, routine application of biomolecular simulations is hampered by the very high computational costs of this method. Several methods have been developed to address this problem. One of them, metadynamics, disfavors states of the simulated system that have been already visited and thus forces the system to explore new states. Here we present the package **metadynminer** and **metadynminer3d** to analyze and visualize results from metadynamics, in particular those produced by a popular metadynamics package Plumed.

1 Introduction

Molecular simulations and their pioneers Martin Karplus, Michael Levitt, and Arieh Warshel have been awarded the Nobel Prize in 2013 (Karplus 2013). Their methods, in particular the method of molecular dynamics simulation, computationally simulate the motions of atoms in a molecular system. A simulation starts from a molecular system defined by positions (Cartesian coordinates) of the individual atoms. The heart of the method is in a calculation of forces acting on individual atoms and their numerical integration in the spirit of Newtonian dynamics, i.e., the conversion of a force vector to an acceleration vector, then velocity vector and, finally, to a new position of an atom. By repeating these steps, it is possible to reconstruct a record of atomic motions known as a trajectory.

Molecular simulations have great potential in drug discovery. A molecule of drug influences (enhances or blocks) the function of some biomolecule in the patient's body, typically a receptor, enzyme or other protein. These molecules are called drug targets. The process of design for a new drug can be significantly accelerated with knowledge of the 3D structure (Cartesian coordinates of atoms) of the target. With such knowledge, it is possible to find a "druggable" cavity in the target and a molecule that fits and favorably binds to this cavity to influence its function. Strong binding implies that the drug influences the target even in low doses, hence does not cause side effects by interacting with unwanted targets.

Experimental determination of the 3D structures of proteins and other biomolecules is a very expensive and laborious process. Molecular simulations can, at least in principle, replace such expensive and laborious experiments by computing. In principle, a molecular simulation starting from virtually any 3D shape of a molecule would end up in energetically the most favorable shape. This is analogous with water flowing from mountains to valleys and not in the opposite way.

Unfortunately, this approach is extremely computationally expensive. The integration step of a simulation must be small enough to comprise the fastest motions in the molecular system. In practical simulations, it is necessary to use femtosecond integration steps. This means that it is necessary to carry out thousands of steps to simulate picoseconds, millions of steps to simulate nanoseconds, and so forth. In each step, it is necessary to evaluate a substantial number of interactions between atoms. As a result, it is possible to routinely simulate nano- to microseconds. Longer simulations require special high-performance computing resources.

Protein folding, i.e., the transition from a quasi-random to the biologically relevant 3D structure, takes place in microseconds for very small proteins and in much longer time scales for pharmaceutically interesting proteins. For this reason, prediction of a 3D structure by molecular simulations is limited to few small and fast folding proteins. For large proteins, it is currently impossible or at least far from being routine.

Several methods have been developed to address this problem. Metadynamics (Laio and Parrinello 2002) uses artificial forces to force the system to explore states that have not been previously explored in the simulation. At the beginning of the simulation, it is necessary to chose some parameters of the system referred to as collective variables. For example, numerically expressed compactness of the protein can be used as a collective variable to accelerate its folding from a noncompact to a compact 3D structure. Metadynamics starts as a usual simulation. After a certain number of steps (typically 500),

the values of the collective variables are calculated and from this moment this state becomes slightly energetically disfavored due to the addition of an artificial bias potential in the shape of a Gaussian hill. After another 500 steps, another hill is added to the bias potential and so forth. These Gaussian hills accumulate until they "flood" some energy minimum and help the system to escape this minimum and explore various other states (Figure 1). In the analogy of water floating from mountains to valleys, metadynamics adds "sand" to fill valleys to make water flow from valleys back to mountains. This makes the simulation significantly more efficient compared to a conventional simulation because the "water" does not get stuck anywhere.

Using the application of metadynamics, it is possible to significantly accelerate the process of folding. Hopefully, by the end of metadynamics we can see folded, unfolded, and many other states of the protein. However, the interpretation of the trajectory is not straightforward. In standard molecular dynamics simulation (without metadynamics), the state which is the most populated is the most populated in reality. This is not true anymore with metadynamics.

Packages **metadynminer** and **metadynminer3d** use the results of metadynamics simulations to calculate the free energy surface of the molecular system. The most favored states (states most populated in reality) correspond to minima on the free energy surface. The state with the lowest free energy is the most populated state in the reality, i.e., the folded 3D structure of the protein.

As an example to illustrate metadynamics and our package, we use an ultrasimple molecule of "alanine dipeptide" (Figure 1). This molecule can be viewed as a "protein" with just one amino acid residue (real proteins have hundreds or thousands of amino acid residues). As a collective variable it is possible to use an angle ϕ defined by four atoms. Biasing of this collective variable accelerates a slow rotation around the corresponding bond. Figure 1 shows the free energy surface of alanine dipeptide as the black thick line. It is not known before the simulation. The simulation starts from the state B. After 500 simulation steps, the hill is added (the hill is depicted as the red line, the flooding potential ("sand") at the top, the free energy surface with added flooding potential at the bottom). The sum of 1, 10, 100, 200, 500, and 700 hills are depicted as red to blue lines.

At the end of simulation the free energy surface is relatively well flattened (blue line in Fig. 1 bottom). Therefore, the free energy surface can be estimated as a negative imprint of added "sand":

$$G(s) = -kT\log(P(s)) = -V(s) = \sum_i w_i \exp(-(s-S_i)^2/2\sigma^2), \tag{1}$$

where G, V, and P are free energy, metadynamics bias (flooding) potential, and probability, respectively, of a state with a collective variable s, k is Boltzmann constant, T is temperature in Kelvins, w_i is height, S_i is position and σ_i is width of each hill. The equation can be easily generalized for two or more collective variables.

The original version of metadynamics was developed with constant heights of Gaussian hills. Later, a so-called well-tempered metadynamics was developed (Barducci, Bussi, and Parrinello 2007), which uses decreasing hill heights to improve the accuracy of the results. This requires modification of the equation:

$$G(s) = -kT\log(P(s)) = -\frac{T+\Delta T}{\Delta T}V(s) = -\frac{T+\Delta T}{\Delta T}\sum_i w_i \exp(-(s-S_i)^2/2\sigma^2), \tag{2}$$

where ΔT an input parameter with the dimension of temperature (zero for unbiased simulation and infinity for the original metadynamics with constant hill heights). Nowadays, the vast majority of metadynamics applications use the well-tempered metadynamics algorithm for better convergence towards an accurate free energy surface prediction.

There are numerous packages for molecular simulations such as Amber (Weiner and Kollman 1981), Gromacs (Abraham et al. 2015), Gromos (Christen et al. 2005), NAMD (Phillips et al. 2020), CHARMM (Brooks et al. 2009), Acemd (Harvey, Giupponi, and Fabritiis 2009), and others. These packages are primarily developed for basic unbiased simulations with no or very limited support of metadynamics. Plumed software (Tribello et al. 2014) has been developed to introduce metadynamics into various simulation programs. Since its introduction, Plumed articles have been cited in more than thousand papers from drug design, molecular biology, material sciences, and other fields. The R package **metadynminer** was developed for analysis and visualization of the results from Plumed. With a simple file conversion script, it can be used also with other simulation programs that support metadynamics.

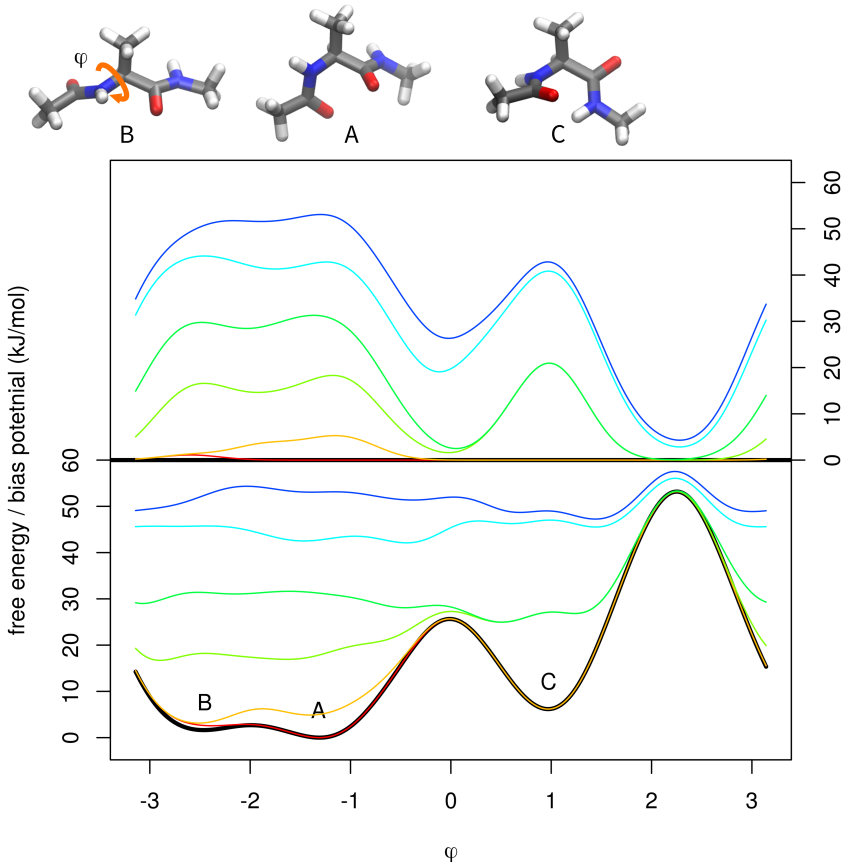


Figure 1: Metadynamics simulation of alanine dipeptide. Dihedral angle ϕ was used as the collective variable. The top part shows molecular structures of three free energy minima (stable structures) differing in the value of ϕ. According to metadynamics prediction, A is the global minimum (free energy 0 kJ/mol) and B and C are local minima (1.5 and 6.3 kJ/mol, respectively). According to Equation 1, this corresponds to probabilities 0.61, 0.34, and 0.05 for A, B, and C, respectively. The middle part shows the bias potential (scaled by $(T + \Delta T)/\Delta T$) after addition of 1, 10, 100, 200, 500, and 700 hills (colors from red to blue). The bottom part shows the accurate free energy surface calculated by metadynamics with 30,000 hills (black) flooded by 1, 10, 100, 200, 500, and 700 hills (colors from red to blue). The figure was generated by metadynminer except for molecular structures and final assembly.

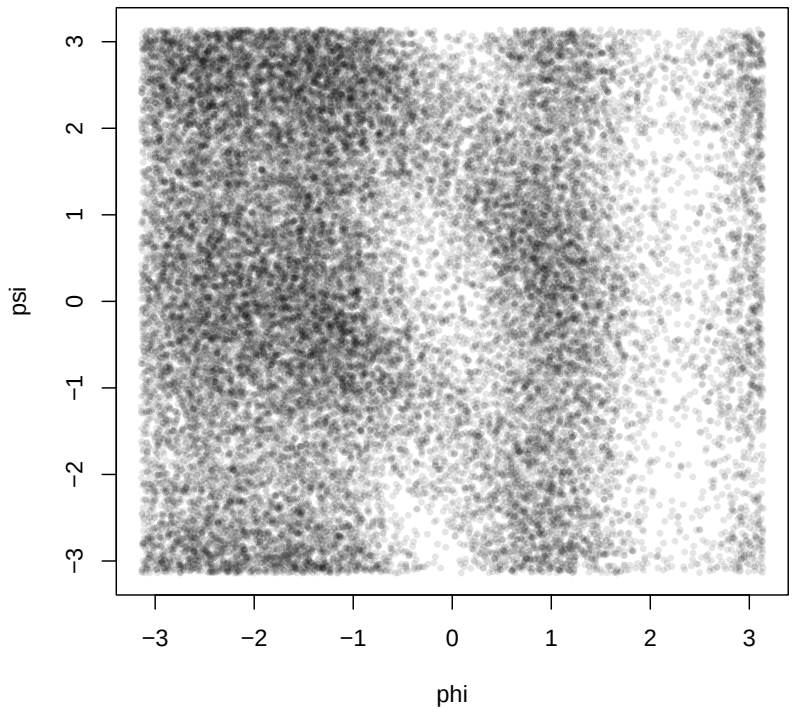


Figure 2: Scatter plot of hills position. Each point in the plot represents a single hill in the space of collective variable coordinates. This helps to assess which states of the system were sampled.

2 Example of usage

The package **metadynminer** will be presented on a bias potential from a 30 ns (30,000 hills) simulation of alanine dipeptide (Figure 1). Two rotatable bonds of the molecule, referred to as ϕ and ψ, were used as collective variables. This is basically an expansion of the free energy surface in Figure 1 to two dimensions. Hills from simulations with two collective variables (ϕ and ψ) and with one collective variable (ϕ) are provided in **metadynminer** as `acealanme` and `acealanme1d`, respectively. **metadynminer3d** was developed for analysis of metadynamics with three collective variables. It contains a sample data `acealanmed3`, with collective variables ϕ, ψ and ω. We decided to distribute **metadynminer** and **metadynminer3d** separately, because of the use of different visualization tools and to keep the size of packages low. Metadynamics simulations with 1-3 collective variables comprise almost all metadynamics applications nowadays (not considering special metadynamics variants).

Hills file generated by Plumed package (filename HILLS) can be loaded to R by the function `read.hills`:

```
hillsfile <- read.hills("HILLS.txt", per=c(TRUE, TRUE))
```

The parameter `per` indicates periodicity of the collective variable (dihedral angles are periodic, i.e., $+\pi \simeq -\pi$). For the simulation described above, `hillsfile` is identical to `acealanme` already contained in **metadynminer** as an example.

Typing the name `hillsfile` will return its dimensionality (the number of collective variables) and the number of hills. A hills object can be plotted:

```
plot(hillsfile, xlab="phi", ylab="psi", pch=19, cex=0.5, col=gray(0, 0.1))
```

For metadynamics with one collective variable, it plots its evolution. For metadynamics with two or three collective variables, it plots a scatter plot of collective variables number 1 vs. 2 or 1 vs. 2 vs. 3, respectively (Figure 2).

In well-tempered metadynamics it may be interesting to see the evolution of hill heights (w_i in Equation (2)). This can be plotted (Figure 3) by typing:

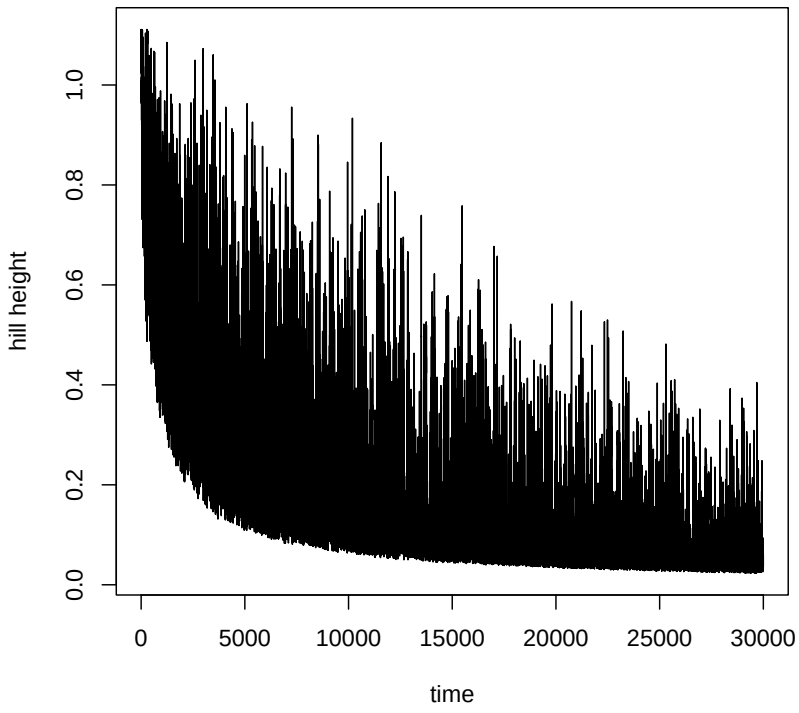


Figure 3: Evolution of heights of hills in metadynamics plotted by function plotheights. In well-tempered metadynamics, heights of hills decrease with the progress of flooding of free energy minima. The evolution of heights of hills may help to assess the completeness of flooding.

```
plotheights(hillsfile)
```

Addition operation is available for hillsfile object. For example, multiple hills files can be concatenated.

Next, the user can sum negative values of all hills to make the free energy surface estimate by typing:

```
fesurface <- fes(hillsfile)
```

Hills files from well-tempered metadynamics are prescaled by $(\Delta T + T)/\Delta T$ when printed by Plumed, so no special action is required in **metadynminer**. The function `fes` uses the Bias Sum algorithm (Hošek and Spiwok 2016). This function is fast because instead of evaluation of Gaussian function for every hill, it uses a precomputed Gaussian hill that is relocated to hill centers. It is also fast because it was implemented in C++ via **Rcpp**. Because of approximations used in the function `fes`, this function should be used for visualization purposes. For detailed analysis of a free energy surface, we advise to use a slow but accurate `fes2` function. This function explicitly evaluates Gaussian function for every hill. It can be also used for (rarely used) metadynamics with variable hill widths.

Typing the name of the variable with a free energy surface returns its dimensionality, number of points, and free energy maximum and minimum. The same is returned by `summary` function. It is possible to add and subtract two free energy surfaces with the same number of grid points. The functions `min` and `max` can be used as well to calculate minimum or maximum. It is also possible to multiply or divide the free energy surface by a constant (for example, to convert kJ to kcal and vice versa). Free energy surface can be plotted (Figure 4) by typing:

```
plot(fesurface, xlab="phi", ylab="psi")
```

In metadynamics simulation, it is important to find free energy minima. The global minimum refers to the most favored state of the system (i.e., the state with the highest probability). Other local minima correspond to metastable states. The user can find free energy minima by typing:

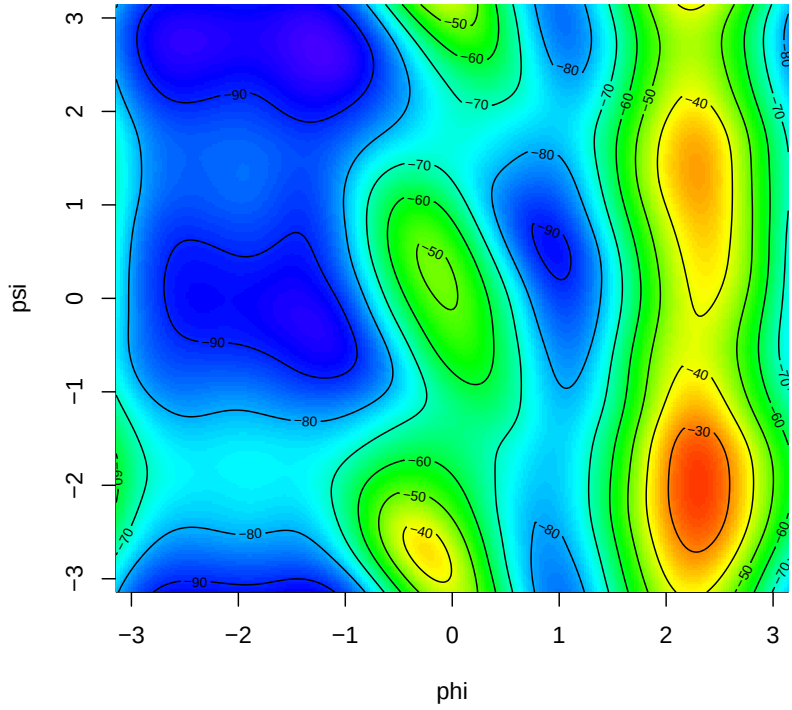


Figure 4: Free energy surface. Minima (blue colors) represent stable states with high abundance, whereas regions with high free energy correspond to low abundance states.

```
minima <- fesminima(fesurface)
```

This function locates minima using a simple algorithm. The free energy surface is separated into 8, 8x8, or 8x8x8 bins (for 1D, 2D, or 3D surface, respectively). The minimum in each bin is located. Next, the program tests whether the minimum is a local minimum of the whole free energy surface. The number of grid points can be changed by `ngrid` parameter. Typing the name of the minima variable will return the table of minima (denoted as A, B, C, ... in the order of their free energies), their collective variables, and free energy values.

In addition, the function summary provides populations of each minimum calculated as:

$$P_{i,rel} = \exp(-G_i/kT), \tag{3}$$

$$P_i = P_{i,rel} / \sum(P_{j,rel}). \tag{4}$$

```
#>   letter CV1bin CV2bin        CV1        CV2 free_energy relative_pop
#> 1      A     78    236 -1.2443171  2.6487938   -97.26095 8.614856e+16
#> 2      B     28    240 -2.4763142  2.7473536   -95.63038 4.480527e+16
#> 3      C     74    118 -1.3428769 -0.2587194   -94.73163 3.124915e+16
#> 4      D    166    151  0.9239978  0.5543987   -91.66626 9.143024e+15
#> 5      E    170    251  1.0225576  3.0183929   -84.37799 4.920882e+14
#>          pop
#> 1 50.1335658
#> 2 26.0741201
#> 3 18.1852268
#> 4  5.3207200
#> 5  0.2863674
```

Using the plot function on a `fesminima` output provides the same plot as for `fes` output with additional letters indicating minima (Figure 5).

It is essential to evaluate the accuracy of metadynamics and to decide when the simulation is accurate enough so that it can be stopped. For this purpose, it is useful to look at the evolution of

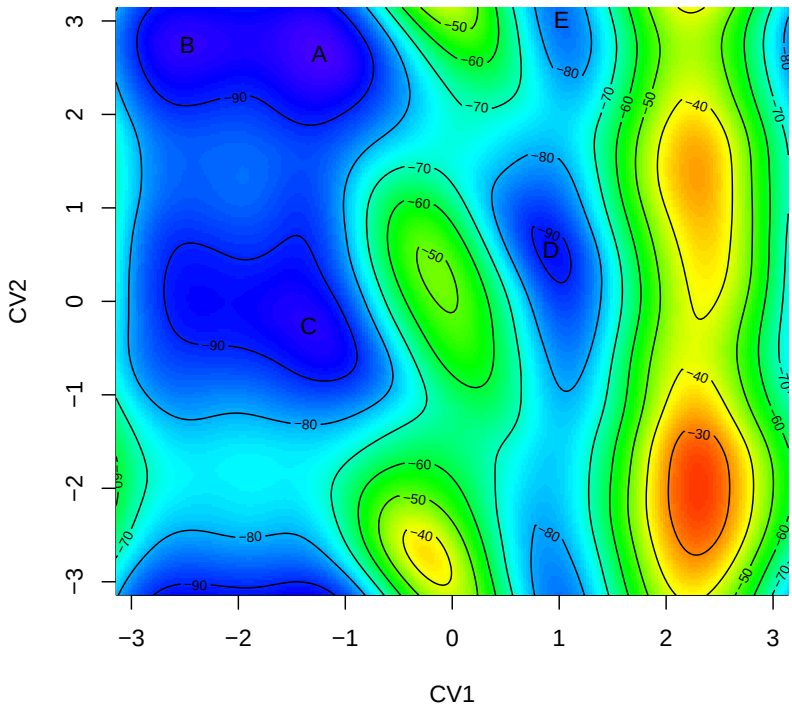


Figure 5: Free energy surface with indicated free energy minima A-E. The minimum A is the most abundant state, minima B-E are metastable states.

relative free energies. The relative free energies (for example, the free energy difference of minima A and C) evolve rapidly at the beginning of the simulation, and with the progress of the simulation, their difference is converging towards the real free energy difference. Function `feprof` calculates the evolution of free energy differences from the global minimum (global at the end of the simulation). It can be used as:

```
prof <- feprof(minima)
```

Function summary provides minima and maxima of these free energy differences. The evolution can be plotted (Figure 6) by typing:

```
plot(prof)
```

Beside minima, another important points on the free energy surface are transition states. Change of the molecular structure from one minimum to another takes place via a path with the lowest energy demand. The state with the highest energy along this path is called the transition state. Free energy difference between the initial and transition state can be used to predict kinetics (rates) of the studied molecular process. Furthermore, identification of transition states is important in drug design because compounds designed to mimic the transition states of an enzymatic reaction are often potent enzymes inhibitors and thus good drug candidates (Itzstein et al. 1993).

In **metadynminer**, such path can be identified by Nudged Elastic Band method (Henkelman and Jónsson 2000). Briefly, this method plots a line between selected minima as an initial approximation of the transition path. Next, this line is curved so that the corresponding physical process becomes feasible. This function can be applied on, for example, minima A and D as:

```
nebAD <- neb(minima, min1="A", min2="D")
```

The result can be analyzed by `summary` (to provide kinetics of the A to D and D to A change predicted by Eyring equation (Eyring 1935)), by `plot` (to plot the free energy profile of the molecular process) and by `pointsonfes` or `linesonfes` (to plot the path on top of the free energy surface). The last example can be invoked by:

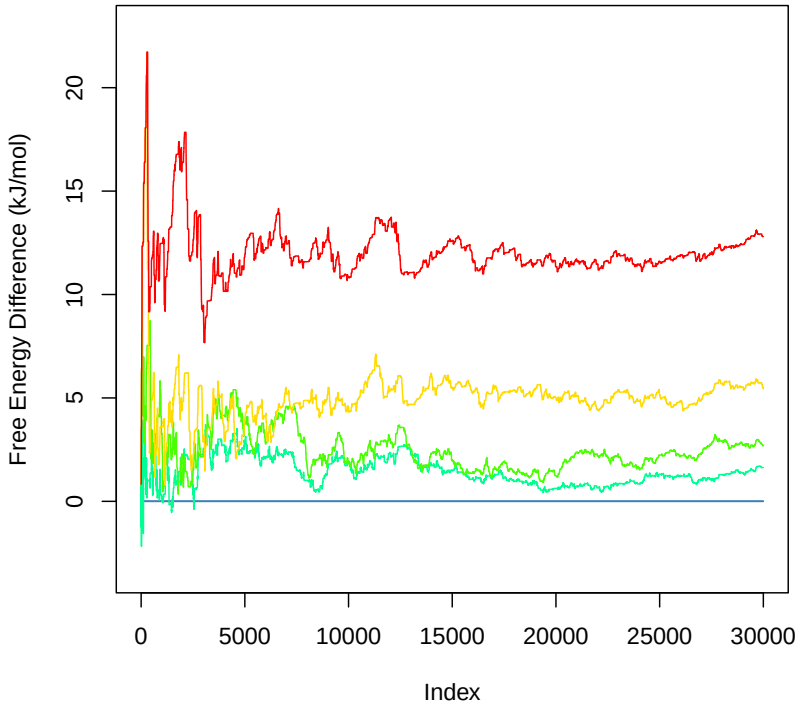


Figure 6: Evolution of free energy differences. The free energy differences of minima B-E (relative to the global minimum A) converge to the exact free energy differences with the progress of the simulation. This plot helps to assess the accuracy of the predicted free energy differences.

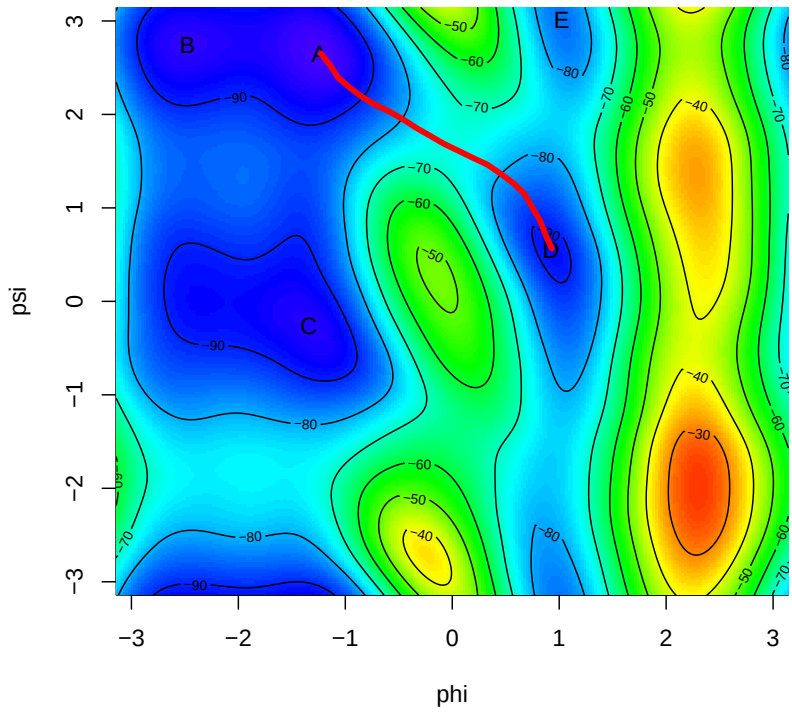


Figure 7: Path of transition from A to D projected onto free energy surface. This represents the most favorable path between these minima. It can be used to identify the transition state (the point with the highest energy on the path) and the rate of the transition.

```
plot(minima, xlab="phi", ylab="psi")
linesonfes(nebAD, lwd=4)
```

The resulting plot is depicted in Figure 7.

Let us also briefly present **metadynminer3d**. This package uses packages **rgl** and **misc3d** to plot the free energy surface as an interactive (mouse rotatable) isosurface in the space of three collective variables (see Figure 8). **metadynminer3d** can produce interactive WebGL visualizations using `writeWebGL` command from the **rgl** package.

metadynminer and **metadynminer3d** were developed to be highly flexible. This flexibility can be demonstrated on two examples. First, it is useful to visualize the progress of metadynamics as a video sequence showing the evolution of the free energy surface. The code to generate corresponding images can be written in **metadynminer** as:

```
tfes <- fes(hillsfile, tmax=100)
png("snap%04d.png")
plot(tfes, zlim=c(-200,0))
for(i in 1:299) {
  tfes <- tfes+fes(acealanme, imin=100*i+1, imax=100*(i+1))
  plot(tfes, zlim=c(-200,0), xlab="phi", ylab="psi")
}
dev.off()
```

This generates a series of images that can be concatenated by external software to make a video file.

The second example demonstrates a more complicated analysis of the results from metadynamics. Functions `fes` and `fes2` use equations (1) and (2) to predict the free energy surface. A limitation of this approach is that the prediction of the free energy surface is based only on the positions of hills. The evolution of collective variables between hills depositions is not used. As an alternative, it is possible to use reweighting (Torrie and Valleau 1977),(Tiwary and Parrinello 2015). This approach

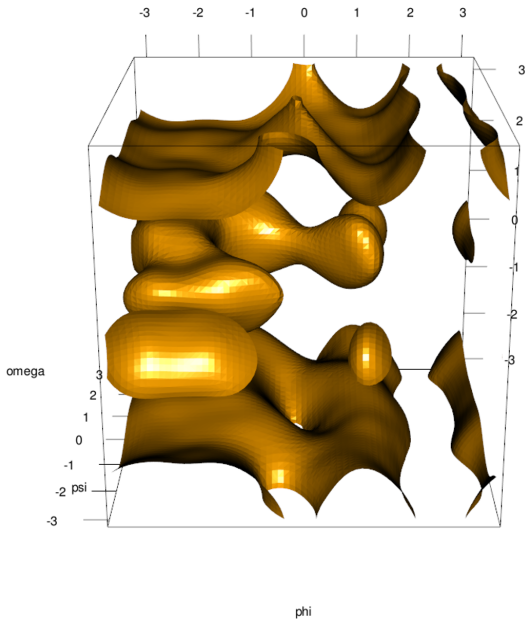


Figure 8: 3D free energy surface depicted as isosurface at -30 kJ/mol. It is analogous to the 2D plot in Figure 4, but with three collective variables.

calculates the free energy surface from hills positions as well as from evolution of collective variables. Briefly, regions of the free energy surface that are sampled despite being disfavored by high flooding potential have higher weights than those disfavored by low flooding potential. This approach, in general, is more accurate. A file containing values of collective variables and the bias potential at different snapshots of the simulation (default filename COLVAR) is required. Reweighting can be done using the code:

```
bf <- 15
kT <- 8.314*300/1000
npoints <- 50
maxfes <- 75
outfes <- 0*fes(hillsfile, npoints=npoints)
step <- 1:50*length(hillsfile$time)/50
s1 <- sapply(step, FUN=function(x) {
          sum(exp(-fes(hillsfile, imax=x)$fes/kT))
})
s2 <- sapply(step, FUN=function(x) {
          sum(exp(-fes(hillsfile, imax=x)$fes/kT/bf))
})
ebetac <- s1/s2
cvs <- read.table("COLVAR.txt")
nsamples <- nrow(cvs)
xlim <- c(-pi,pi)
ylim <- c(-pi,pi)
step <- (1:nsamples-1)*50/nsamples+1
ix <- npoints*(cvs[,2]-xlim[1])/(xlim[2]-xlim[1])+1
iy <- npoints*(cvs[,3]-ylim[1])/(ylim[2]-ylim[1])+1
for(i in 1:nsamples) {
  outfes$fes[ix[i],iy[i]] <- outfes$fes[ix[i],iy[i]] + exp(cvs[i,4]/kT)/
ebetac[step[i]]
}
outfes$fes <- -kT*log(outfes$fes)
```

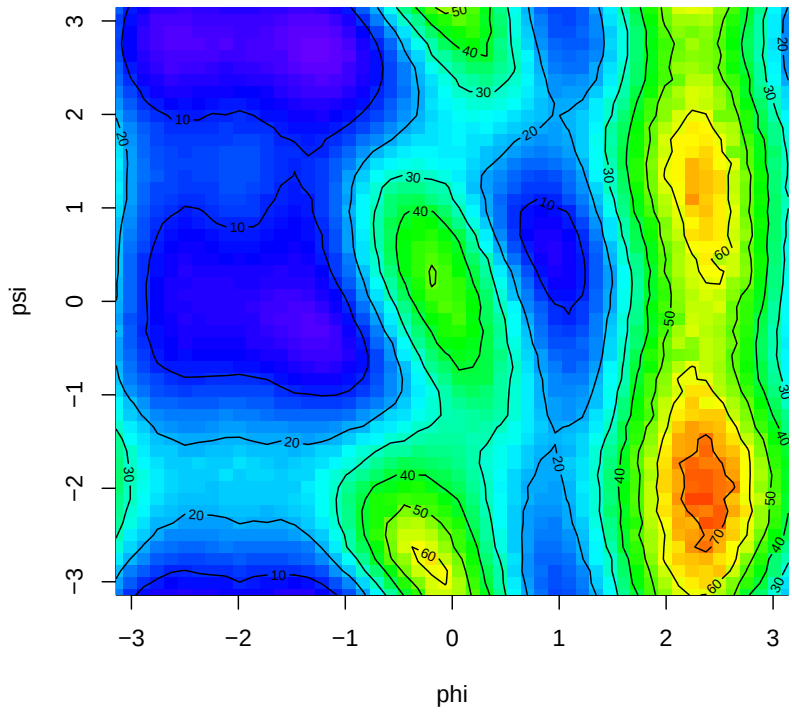


Figure 9: Free energy surface calculated by reweighting by Tiwary and Parrinello. This free energy surface was calculated by combining the information on time spent in different regions of the free energy surface and on the potential disfavoring these regions. This approach is in general more accurate than the summation of hills used to generate Figures 4-8.

```
outfes <- outfes - min(outfes)
outfes$fes[outfes$fes>maxfes] <- maxfes
plot(outfes, xlab="phi", ylab="psi")
```

where `bf` is the bias factor ($(T+\Delta T)/T$ in Equation (2)), `kT` is temperature in Kelvins multiplied by Boltzmann constant, `npoints` is the granularity of the resulting free energy surface and `maxfes` is the maximal possible free energy (to avoid problems with infinite free energy in unsampled regions). First, `outfes` is introduced as a zero free energy surface. First, the correction `ebetac` for the evolution of flooding potential developed by Tiwary and Parrinello (Tiwary and Parrinello 2015) is calculated. Next, a file with the evolution of collective variables COLVAR (from the same simulation used to generate `acealanme` dataset, available at https://www.metadynamics.cz/metadynminer/data/) is read. The second loop evaluates the sampling weighted by the factor $\exp(V(s)/kT)$ divided by `ebetac` to correct for the evolution of the bias potential (Tiwary and Parrinello 2015). Finally, probabilities are converted to the free energy surface and plotted (Figure 9).

3 Simulation details

All simulations were done using Gromacs 2016.4 (Abraham et al. 2015) patched by Plumed 2.4b (Tribello et al. 2014). Alanine dipeptide was modeled using Amber99SB-ILDN force field (Lindorff-Larsen et al. 2010). The simulated system contained alanine dipeptide and 874 TIP3P (Jorgensen et al. 1983) water molecules. The temperature was kept constant at 300 K using Bussi thermostat (Bussi, Donadio, and Parrinello 2007). Metadynamics hills of height 1 kJ/mol (bias factor 10) and widths 0.3 rad were added every 1 ps. Two simulations were done, one with one dihedral angle ϕ (dataset `acealanme1d`), two dihedral angles ϕ and ψ (dataset `acealanme`), or with three angle ϕ, ϕ and ω (dataset `acealanme3d` in **metadynminer3d**). Supporting material is available at https://www.metadynamics.cz/metadynminer/data/ or in Plumed nest (PLUMED consortium 2019) at https://www.plumed-nest.org/eggs/20/023/.

4 Summary

The package **metadynminer** and **metadynminer3d** provides fast algorithm Bias Sum (Hošek and Spiwok 2016) for calculation of free energy surfaces from metadynamics. This algorithm is available in our on-line tool MetadynView (http://metadyn.vscht.cz), but this tools is intended for routine checking of the progress of metadynamics simulations rather than for in-depth analysis and visualization. Besides this, users of metadynamics use built-in functions in Plumed or various in-lab scripts. Such scripts do not provide appropriate flexibility in analysis and visualization.

The biggest advantage we see is in the fact that **metadynminer** can produce publication quality figures via graphics output functions in R. As shown above, using a simple `for` loop it is possible to plot individual snapshots and concatenate them outside R to make a movie. **metadynminer3d** provides the possibility to produce interactive 3D web models by WebGL technology. We also tested 3D printing of a free energy surface that is very easy using **metadynminer** and **rayshader**. Various tips and tricks can be found on the website of the project (https://www.metadynamics.cz/metadynminer/).

Another advantage we see is in the reporting of results. Reproducibility is a big issue in science, including molecular simulations. Packages like **knitr** or **rmarkdown** ca be used to record all steps of data analysis pipeline to compile a report for routine and reproducible use of metadynamics.

5 Acknowledgement

This project was supported by Ministry of Education, Youth and Sports of the Czech Republic - COST action OpenMultiMed (CA15120, LTC18074) for development and Czech National Infrastructure for Biological data (ELIXIR CZ, LM2015047, LM2018131) for future sustainability.

References

Abraham, James Mark, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Lindahl Erik. 2015. "GROMACS: High Performance Molecular Simulations Through Multi-Level Parallelism from Laptops to Supercomputers." *SoftwareX* 1–2: 19–25. https://doi.org/10.1016/j.softx.2015.06.001.

Barducci, Alessandro, Giovanni Bussi, and Michele Parrinello. 2007. "Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method." *Physical Review Letters* 100 (2): 020603. https://doi.org/10.1103/PhysRevLett.100.020603.

Brooks, B. R., C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, et al. 2009. "CHARMM: The Biomolecular Simulation Program." *Journal of Computational Chemistry* 30 (10): 1545–1614. https://doi.org/10.1002/jcc.21287.

Bussi, Giovanni, Davide Donadio, and Michele Parrinello. 2007. "Canonical Sampling Through Velocity Rescaling." *Journal of Chemical Physics* 1 (126): 014101. https://doi.org/10.1063/1.2408420.

Christen, Markus, Philippe H. Hünenberger, Dirk Bakowies, Riccardo Baron, Roland Bürgi, Daan P. Geerke, Tim N. Heinz, et al. 2005. "The GROMOS Software for Biomolecular Simulation: GROMOS05." *Journal of Computational Chemistry* 26 (16): 1719–51. https://doi.org/10.1002/jcc.20303.

Eyring, Henry. 1935. "The Activated Complex in Chemical Reactions." *The Journal of Chemical Physics* 3 (2). https://doi.org/10.1063/1.1749604.

Harvey, M. J., G. Giupponi, and G. De Fabritiis. 2009. "ACEMD: Accelerating Biomolecular Dynamics in the Microsecond Time Scale." *Journal of Chemical Theory and Computation* 5 (6): 1632–39. https://doi.org/10.1021/ct9000685.

Henkelman, Graeme, and Hannes Jónsson. 2000. "Improved Tangent Estimate in the Nudged Elastic Band Method for Finding Minimum Energy Paths and Saddle Points." *The Journal of Chemical Physics* 113 (22): 9978–85. https://doi.org/10.1063/1.1323224.

Hošek, Petr, and Vojtěch Spiwok. 2016. "Metadyn View: Fast Web-Based Viewer of Free Energy Surfaces Calculated, by Metadynamics." *Computer Physics Communications* 198: 222–29. https://doi.org/10.1016/j.cpc.2015.08.037.

Itzstein, Mark von, Wen-Yang Wu, Gaik B. Kok, Michael S. Pegg, Jeffrey C. Dyasson, Betty Jin, Tho Van Phan, et al. 1993. "Rational Design of Potent Sialidase-Based Inhibitors of Influenza Virus Replication." *Nature* 363: 418–23. https://doi.org/10.1038/363418a0.

Jorgensen, William L., Jayaraman Chandrasekhar, Jeffry D. Madura, Roger W. Impey, and Michael L. Klein. 1983. "Comparison of Simple Potential Functions for Simulating Liquid Water." *The Journal of Chemical Physics* 79 (2): 926. https://doi.org/10.1063/1.445869.

Karplus, Martin. 2013. "Nobel Lecture." https://www.nobelprize.org/prizes/chemistry/2013/karplus/lecture/.

Laio, Alessandro, and Michele Parrinello. 2002. "Escaping Free-Energy Minima." *Proceedings of the National Academy of Sciences of the United States of America* 20 (99): 12562–66. https://doi.org/10.1073/pnas.202427399.

Lindorff-Larsen, Kresten, Stefano Piana, Kim Palmo, Paul Maragakis, John L. Klepeis, Ron O. Dror, and David E. Shaw. 2010. "Improved Side-Chain Torsion Potentials for the Amber ff99SB Protein Force Field." *Proteins: Structure, Function, and Bioinformatics* 78 (8): 1950–58. https://doi.org/10.1002/prot.22711.

Phillips, James C., David J. Hardy, Julio D. C. Maia, John E. Stone, João V. Ribeiro, Rafael C. Bernardi, Ronak Buch, et al. 2020. "Scalable Molecular Dynamics on CPU and GPU Architectures with NAMD." *Journal of Chemical Physics* 153 (4): 044130. https://doi.org/10.1063/5.0014475.

PLUMED consortium. 2019. "Promoting Transparency and Reproducibility in Enhanced Molecular Simulations." *Nature Methods* 16: 670–73. https://doi.org/10.1038/s41592-019-0506-8.

Tiwary, Pratyush, and Michele Parrinello. 2015. "A Time-Independent Free Energy Estimator for Metadynamics." *The Journal of Physical Chemistry B* 119 (3): 736–42. https://doi.org/10.1021/jp504920s.

Torrie, G. M., and J. P. Valleau. 1977. "Nonphysical Sampling Distributions in Monte Carlo Free-Energy Estimation: Umbrella Sampling." *Journal of Computational Physics* 23: 187–99. https://doi.org/10.1016/0021-9991(77)90121-8.

Tribello, Gareth A., Massimiliano Bonomi, Davide Branduardi, Carlo Camilloni, and Giovanni Bussi. 2014. "PLUMED 2: New Feathers for an Old Bird." *Computer Physics Communications* 185 (2): 604–13. https://doi.org/10.1016/j.cpc.2013.09.018.

Weiner, Paul K., and Peter A. Kollman. 1981. "AMBER: Assisted Model Building with Energy Refinement. A General Program for Modeling Molecules and Their Interactions." *Journal of Computational Chemistry* 2 (3): 287–303. https://doi.org/10.1002/jcc.540020311.

Dalibor Trapl
Department of Biochemistry and Microbiology, University of Chemistry and Technology, Prague
Technicka 3
Prague 6, 166 28, Czech Republic
ORCiD: 0000-0002-3435-5841
dalibor.trapl@gmail.com

Vojtech Spiwok
Department of Biochemistry and Microbiology, University of Chemistry and Technology, Prague
Technicka 3
Prague 6, 166 28, Czech Republic
ORCiD: 0000-0001-8108-2033
spiwokv@vscht.cz

casebase: An Alternative Framework for Survival Analysis and Comparison of Event Rates

by Sahir Rai Bhatnagar, Maxime Turgeon*, Jesse Islam, James A. Hanley, and Olli Saarela*

Abstract In clinical studies of time-to-event data, a quantity of interest to the clinician is their patient's risk of an event. However, methods relying on time matching or risk-set sampling (including Cox regression) eliminate the baseline hazard from the estimating function. As a consequence, the focus has been on reporting hazard ratios instead of survival or cumulative incidence curves. Indeed, reporting patient risk or cumulative incidence requires a separate estimation of the baseline hazard. Using case-base sampling, Hanley & Miettinen (2009) explained how parametric hazard functions can be estimated in continuous-time using logistic regression. Their approach naturally leads to estimates of the survival or risk function that are smooth-in-time. In this paper, we present the **casebase** R package, a comprehensive and flexible toolkit for parametric survival analysis. We describe how the case-base framework can also be used in more complex settings: non-linear functions of time and non-proportional hazards, competing risks, and variable selection. Our package also includes an extensive array of visualization tools to complement the analysis. We illustrate all these features through three different case studies. * SRB and MT contributed equally to this work.

1 Introduction

The semiparametric Cox model has become the default approach to survival analysis even though Cox himself later suggested he would prefer to model the hazard function directly. In a 1994 interview with Professor Nancy Reid, Sir David Cox was asked how he would model a set of censored survival data, to which he responded: "I think I would normally want to tackle problems parametrically ... and if you want to do things like predict the outcome for a particular patient, it's much more convenient to do that parametrically" (Reid 1994). Indeed, the most relevant quantity in a clinical setting is often the 5- or 10-year risk of experiencing a certain event given the patient's particular profile. However, the most reported metric from a Cox model is the (potentially time-dependent) hazard ratio (HR). The covariate-conditional survival curve is arguably a more important summary measure to report than the HR. While stepwise survival curves can be computed with the Cox model, they require a second step to separately estimate the baseline hazard (Breslow 1972).

Several authors have since pursued fully parametric approaches that made the fitting of smooth survival curves more transparent and intuitive through generalized linear models. The key feature of these procedures is splitting the time axis into discrete intervals. Whitehead (1980) showed the equivalence between a Cox model and a Poisson regression with a parameter for each event time; Carstensen (2019) provides an exposition of this equivalence with a real data example and supporting R code for computing standard errors. Arjas & Haara (1987) and Efron (1988) treated each patient-day as a Bernoulli random variable with probability equal to the discrete hazard rate. A potential issue with these approaches is that the number of time bins need to be chosen by the data analyst. On the one hand, a fine grouping of times may result in few (or none) events per interval, which then leads to instability in the Newton-Raphson procedure for estimation (Kalbfleisch and Prentice 2011, vol. 360, sec. 4.8), and large long-format datasets. On the other hand, a coarse grouping could potentially mask nonlinear trends in the hazard function.

Rather than discretizing time, Hanley & Miettinen (2009) selected a discrete set of person-time coordinates ("person-moments") in continuous time from all observed follow-up experience constituting the study base. By doing so, they obtained a likelihood expression for the hazard function that is equivalent to that of logistic regression with an offset term. More specifically, all person-moments when the event of interest occurred are selected as the case series, complemented by a randomly sampled base series of person-moments serving as controls. This approach allows flexible modeling of the hazard function by including functions of time as covariates (e.g. using splines or generalized additive models). Furthermore, time-dependent covariates can be modeled through interactions with time. In short, Hanley & Miettinen (2009) use the well-understood logistic regression for directly modeling the hazard function, without requiring a discrete-time model.

In this article, we present the **casebase** R package (Bhatnagar et al. 2021) which implements and extends the Hanley & Miettinen (2009) approach for fitting fully parametric hazard models and covariate-conditional survival curves using the familiar interface of the `glm` function. Our implementation includes extensions to other models such as penalized regression for variable selection

and competing risk analysis. In addition, we provide functions for exploratory data analysis and visualizing the estimated quantities such as the hazard function, survival curve, and their standard errors. The ultimate goal of our package is to make fitting flexible hazards accessible to end users who favor reporting absolute risks and survival curves over hazard ratios.

In what follows, we first recap some theoretical details on case-base sampling and its use for estimating parametric hazard functions. We then give a short review of existing R packages that implement comparable features to **casebase**. Next, we provide some details about the implementation of case-base sampling in our package, and we give a brief survey of its main functions. This is followed by three case studies that illustrate the flexibility and capabilities of **casebase**. We show how the same framework can be used for non-linear functions of time and non-proportional hazards, competing risks, and variable selection via penalized regression. Finally, we end the article with a discussion of the results and of future directions.

2 Theoretical details

As discussed in Hanley & Miettinen (2009), the key idea behind case-base sampling is to consider the entire study base as an infinite collection of *person-moments*. These person-moments are indexed by both an individual in the study and a time point, and therefore each person-moment has a covariate profile and an outcome status (i.e. whether the event happened) attached to it. By estimating the probability of an event occurring at a particular person-moment, we can extract information about hazards and survival.

Therefore, we start by assembling all person-moments at which the event occurred; this collection of person-moments is what Hanley & Miettinen call the *case series*. The incidence of the case series is dictated by the hazard function of interest. Next, we sample a finite number of person-moments (blinded to case moments); this second collection of person-moments is what Hanley & Miettinen call the *base series*. The sampling mechanism for the base series is left at the discretion of the user, but in practice we find that sampling uniformly from the study base provides both simplicity and good performance. This is the default sampling mechanism in the package.

Likelihood and estimating function

To describe the theoretical foundations of case-base sampling, we use the framework of counting processes. In what follows, we abuse notation slightly and omit any mention of σ-algebras. Instead, following Aalen *et al* (2008), we use the placeholder "past" to denote the past history of the corresponding process. The reader interested in more details can refer to Saarela & Arjas (2015) and Saarela (2016). First, let $N_i(t) \in \{0, 1\}$ be counting processes corresponding to the event of interest for individual $i = 1, \ldots, n$. For simplicity, we will consider Type I censoring due to the end of follow-up at time τ (the general case of independent censoring is treated in Saarela (2016)). We are interested in modeling the hazard functions $\lambda_i(t)$ of the processes $N_i(t)$, and which satisfy

$$\lambda_i(t)\,\mathrm{d}t = E[\,\mathrm{d}N_i(t) \mid \text{past}].$$

The processes $N_i(t)$ count the person-moments from the *case series*.

To complement the case series, we sample person-moments for the base series. To do so, we specify the base series sampling mechanism as non-homogeneous Poisson processes $R_i(t) \in \{0, 1, 2, \ldots\}$; the person-moments where $\mathrm{d}R_i(t) = 1$ constitute the base series. We note that the same individual can contribute multiple person-moments to the base series. The process $Q_i(t) = R_i(t) + N_i(t)$ then counts both the case and base series person-moments contributed by individual i. As mentioned above, the processes $R_i(t)$ are specified by the user via its intensity function $\rho_i(t)$. The process $Q_i(t)$ is characterized by $E[\,\mathrm{d}Q_i(t) \mid \text{past}] = \lambda_i(t)\,\mathrm{d}t + \rho_i(t)\,\mathrm{d}t$.

If the hazard function $\lambda_i(t;\theta)$ is parametrized in terms of θ, we can define an estimator $\widehat{\theta}$ by maximization of the likelihood expression

$$L_0(\theta) = \prod_{i=1}^{n} \exp\left\{-\int_0^{\min(t_i,\tau)} \lambda_i(t;\theta)\,\mathrm{d}t\right\} \prod_{i=1}^{n} \prod_{t\in[0,\tau)} \lambda_i(t;\theta)^{\,\mathrm{d}N_i(t)},$$

where $\prod_{t\in[0,u)}$ represents a product integral from 0 to u, and where t_i is the event time for individual i. However, the integral over time makes the computation and maximization of $L_0(\theta)$ challenging using standard software.

Case-base sampling allows us to avoid this integral. By conditioning on a sampled person-moment,

we get individual partial likelihood contributions of the form

$$P(\,\mathrm{d}N_i(t) \mid \mathrm{d}Q_i(t) = 1, \text{past}) \overset{\theta}{\propto} \frac{\lambda_i(t;\theta)^{\,\mathrm{d}N_i(t)}}{\rho_i(t) + \lambda_i(t;\theta)}.$$

By using the symbol $\overset{\theta}{\propto}$, we mean that both sides of the expression are equal up to multiplicative factors that do not depend on θ. Therefore, an estimating function for θ can be composed of these contributions as:

$$L(\theta) = \prod_{i=1}^{n} \prod_{t \in [0,\tau)} \left(\frac{\lambda_i(t;\theta)^{\,\mathrm{d}N_i(t)}}{\rho_i(t) + \lambda_i(t;\theta)} \right)^{\mathrm{d}Q_i(t)}. \tag{1}$$

When a logarithmic link function is used for modeling the hazard function, the above expression is of a logistic regression form with an offset term $\log(1/\rho_i(t))$. Note that the sampling units selected in the case-base sampling mechanism are person-moments, rather than individuals, and the parameters to be estimated are hazards or hazard ratios rather than odds or odds ratios. Generally, an individual can contribute more than one person-moment, and thus the terms in the product integral are not independent. Nonetheless, Saarela (2016) showed that the corresponding partial likelihood score function has mean zero at the true value $\theta = \theta_0$, and that the resulting estimator $\widehat{\theta}$ is asymptotically normally distributed.

In Hanley & Miettinen (2009), the authors suggest sampling the base series *uniformly* from the study base. In terms of Poisson processes, their sampling strategy corresponds to a time-homogeneous Poisson process with intensity equal to $\rho_i(t) = b/B$, where b is the number of sampled base series *person-moments*, and B is the total population-time for the study base (e.g. the sum of all individual follow-up times). More complex examples are also possible; see for example Saarela & Arjas (2015), where the intensity functions for the sampling mechanism are proportional to the cardiovascular disease event rate given by the Framingham score. Non-uniform sampling mechanisms can increase the efficiency of the resulting maximum partial likelihood estimators.

Variance estimates

The asymptotic normality of $\widehat{\theta}$ gives us an efficient way to estimate the variance of various estimators derived from the hazard function. For example, to construct confidence bands around risk functions and survival functions, we can use the following procedure:

1. Compute $\widehat{\theta}$ and its variance-covariance matrix $V(\widehat{\theta})$ which is obtained as part of the logistic regression output.
2. Sample B times from a multivariate normal $N\left(\widehat{\theta}, V(\widehat{\theta})\right)$.
3. For each sample, compute the survival function using Equation (6) below.
4. Use the pointwise quantiles of these survival function estimates to construct a pointwise confidence band for the survival function of interest.

This procedure is similar to parametric bootstrap (Efron and Tibshirani 1994, sec. 6.5), but it can be more accurately described as a form of approximate Bayesian computation. As such, the validity of the confidence bands relies on the Bernstein-von Mises theorem (Van der Vaart 2000, vol. 3, chap. 10).

Common parametric models

Here we show that certain named distributions such as exponential, Weibull or Gompertz can be fit using our framework, though we are not restricted to such models. Let $g(t;X)$ be the linear predictor such that $\log(\lambda(t;X)) = g(t;X)$. Different functions of t lead to different parametric hazard models. The simplest of these models is the one-parameter exponential distribution which is obtained by taking the hazard function to be constant over the range of t:

$$\log(\lambda(t;X)) = \beta_0 + \beta_1 X. \tag{2}$$

In this model, the instantaneous failure rate is independent of t.[1]

The Gompertz hazard model is given by including a linear term for time:

$$\log(\lambda(t;X)) = \beta_0 + \beta_1 t + \beta_2 X. \tag{3}$$

[1]The conditional chance of failure in a time interval of specified length is the same regardless of how long the individual has been in the study. This is also known as the *memoryless property* (Kalbfleisch and Prentice 2011).

Use of $\log(t)$ yields the Weibull hazard which allows for a power dependence of the hazard on time (Kalbfleisch and Prentice 2011):

$$\log(\lambda(t; X)) = \beta_0 + \beta_1 \log(t) + \beta_2 X. \tag{4}$$

Competing risk analysis

Case-base sampling can also be used in the context of competing risk analysis. Assuming there are J competing events, we can show that each sampled person-moment's contribution to the partial likelihood is of the form

$$\frac{\lambda_j(t)^{\,\mathrm{d}N_j(t)}}{\rho(t) + \sum_{j=1}^{J} \lambda_j(t)},$$

where $N_j(t)$ is the counting process associated with the event of type j and $\lambda_j(t)$ is the corresponding cause-specific hazard function. As may be expected, this functional form is similar to the terms appearing in the likelihood function for multinomial regression with an offset term.[2]

Variable selection

To perform variable selection on the regression parameters $\theta \in \mathbb{R}^p$ of the hazard function, we can add a penalty to the likelihood and optimise the following equation:

$$\min_{\theta \in \mathbb{R}^p} \; -\ell(\theta) + \sum_{j=1}^{p} w_j p_{\lambda,\alpha}(\theta_j) \tag{5}$$

where $\ell(\theta) = \log L(\theta)$ is the log of the likelihood function given in (1), $p_{\lambda,\alpha}(\theta_j)$ is a penalty term controlled by the non-negative regularization parameters λ and α, and w_j is the penalty factor for the jth covariate. These penalty factors serve as a way of allowing parameters to be penalized differently. For example, we could set the penalty factor for time to be 0 to ensure it is always included in the selected model.

3 Comparison with existing packages

Survival analysis is an important branch of applied statistics and epidemiology. Accordingly, there is already a vast ecosystem of R packages implementing different methodologies. In this section, we describe how the functionalities of **casebase** compare to these packages.

At the time of writing, a cursory examination of CRAN's *Survival* Task View reveals that there are over 250 packages related to survival analysis (Allignol and Latouche 2019). For the purposes of this article, we restricted our review to packages that implement at least one of the following features: parametric modeling, non-proportional hazard models, competing risk analysis, penalized estimation, and Cumulative Incidence (CI) estimation. By searching for appropriate keywords in the `DESCRIPTION` file of these packages, we found 60 relevant packages. These 60 packages were then manually examined to determine which ones are comparable to **casebase**. In particular, we excluded packages that were focused on a different set of problems, such as frailty and multistate models. The remaining 14 packages appear in Table 1, along with some of the functionalities they offer.

Parametric survival models are implemented in several packages, each differing in the parametric distributions available: **CFC** (2019), **flexsurv** (2016), **SmoothHazard** (2017), **rstpm2** (2019), **mets** (2014), and **survival** (2015). For example, **SmoothHazard** is limited to Weibull distributions (2017), whereas both **flexsurv** and **survival** allow users to supply any distribution of their choice. **flexsurv**, **SmoothHazard**, **mets** and **rstpm2** can model the effect of time using splines, which allows flexible modeling of the hazard function. As discussed above, **casebase** can model any parametric family whose log-hazard can be expressed as a linear combination of covariates (including time). Therefore, our package is more general in that it allows the user to model any linear or non-linear transformation of time including splines and higher order polynomials. Also, by including interaction terms between

[2] Specifically, it corresponds to the following parametrization for a multinomial random variable Y:

$$\log\left(\frac{P(Y = j \mid X)}{P(Y = J \mid X)}\right) = X^T \beta_j + log(1/\rho), \qquad j = 1, \ldots, J-1.$$

Table 1: Comparison of various R packages for survival analysis. **Competing Risks**: whether an implementation for competing risks is present. **Allows Non PH**: includes models for non-proportional hazards. **Penalized Regression**: allows for a penalty term on the regression coefficients when estimating hazards (e.g. lasso or ridge). **Splines**: allows a flexible fit on time through the use of splines. **Parametric**: implementation for parametric models. **Semi-parametric**: implementation for semi-parametric models. **Interval/left censoring**: models for interval and left-censoring. If this is not selected, the package only handles right-censoring. **Risk estimates**: estimation of survival curve and cumulative incidence is available.

Package	Competing Risks	Allows Non PH	Penalized Regression	Splines	Parametric	Semi Parametric	Interval/Left Censoring	Risk Estimates
casebase	X	X	X	X	X			X
CFC	X	X			X			X
cmprsk	X					X		X
crrp	X		X			X		
fastcox			X			X		
flexrsurv		X		X	X			X
flexsurv	X	X		X	X			X
glmnet			X			X		X
glmpath			X			X		
mets	X			X		X		X
penalized			X			X		
riskRegression	X		X			X		X
rstpm2		X		X	X	X	X	X
SmoothHazard		X		X	X		X	
survival	X	X			X	X	X	X

covariates and time, it also allows users to fit (non-proportional) time-varying coefficient models. However, unlike **flexsurv**, we do not explicitly model any shape parameter.

Several packages implement penalized estimation for the Cox model: **glmnet** (2011), **glmpath** (2018), **penalized** (2010), **riskRegression** (2020). Moreover, some packages also include penalized estimation in the context of Cox models with time-varying coefficients: elastic-net penalization with **rstpm2** (2019), while **survival** (2015) has an implementation of ridge-penalized estimation. On the other hand, our package **casebase** provides penalized estimation of the hazard function. To our knowledge, **casebase** and **rstpm2** are the only packages to offer this functionality.

Next, several R packages implement methodologies for competing risk analysis; for a different perspective on this topic, see Mahani & Sharabiani (2019). The package **survival** provides functionality for competing risk analysis and multistate modelling. The package **cmprsk** provides methods for cause-specific subdistribution hazards, such as in the Fine-Gray model (1999). On the other hand, the package **CFC** estimates cause-specific CIs from unadjusted, non-parametric survival functions. Our package **casebase** also provides functionalities for competing risk analysis by estimating parametrically the cause-specific hazards. From these quantities, we can then estimate the cause-specific CIs.

Finally, several packages include functions to estimate the survival function and the CI. The corresponding methods generally fall into two categories: transformation of the estimated hazard function, and semi-parametric estimation of the baseline hazard. The first category broadly corresponds to parametric survival models, where the full hazard is explicitly modeled. Using this estimate, the survival function and the CI can be obtained using their functional relationships (see Equations (6) and (7) below). Packages providing this functionality include **CFC**, **flexsurv**, **mets**, and **survival**. Our package **casebase** also follows this approach for both single-event and competing risk analyses. The second category outlined above broadly corresponds to semi-parametric models. These models do not model the full hazard function, and therefore the baseline hazard needs to be estimated separately in order to estimate the survival function. This is achieved using semi-parametric estimators (e.g. Breslow's estimator) or parametric estimators (e.g. spline functions). Packages that implement this approach include **riskRegression**, **rstpm2**, **survival**, and **glmnet**. As mentioned in the introduction, a key distinguishing factor between these two approaches is that the first category leads to smooth estimates of the survival function, whereas the second category often produces estimates in the form of stepwise functions.

4 Implementation details

The functions in the **casebase** package can be divided into two categories: 1) exploratory data analysis, in the form of population-time plots; and 2) parametric modeling of the hazard function. We strove for compatibility with both `data.frames` and `data.tables`; this can be seen in the coding choices we made and the unit tests we wrote.

Population-time plots

Population-time plots are a descriptive visualization of incidence density, where the population time that constitutes the study base is represented by area, and events by points within the area. The case-base sampling approach described above can be visualized in the form of a population time plot. These plots are informative graphical displays of survival data and should be one of the first steps in an exploratory data analysis. The `popTime` function and `plot` method facilitate this task:

1. The `casebase::popTime` function takes as input the original dataset along with the column names corresponding to the timescale, the event status and an exposure group of interest (optional). This will create an object of class `popTime`.

2. The corresponding `plot` method for the object created in Step 1 can be called to create the population time plot with several options for customizing the aesthetics.

By splitting these tasks, we give flexibility to the user. While the method call in Step 2 allows further customization by using the **ggplot2** (Wickham 2016) family of functions, users may choose the graphics system of their choice to create population-time plots from the object created in Step 1.

To illustrate these functions, we will use data from the European Randomized Study of Prostate Cancer Screening (ERSPC) (Schröder et al. 2009) which was extracted using the approach described in Liu *et al.* (2014). This dataset is available through the **casebase** package. It contains the recreated individual observations for 159,893 men from seven European countries, who were between the ages of 55 and 69 years when recruited for the trial.

We first create the necessary dataset for producing the population time plot using the `popTime` function. In this example, we stratify the plot by treatment group. The resulting object inherits from class `popTime` and stores the exposure variable as an attribute:

```
pt_object <- casebase::popTime(ERSPC, time = "Follow.Up.Time",
                               event = "DeadOfPrCa", exposure = "ScrArm")
inherits(pt_object, "popTime")
#> [1] TRUE
attr(pt_object, "exposure")
#> [1] "ScrArm"
```

We then pass this object to the corresponding `plot` method:

```
plot(pt_object, add.base.series = TRUE)
```

Figure 1 depicts the process of creating a population-time plot. It is built sequentially by first adding a layer for the area representing the population time in gray (Figure 1A), with subjects having the least amount of observation time plotted at the top of the y-axis. We immediately notice a distinctive *stepwise shape* in the population time area. This is due to the randomization of the Finnish cohorts which were carried out on January 1 of each of year from 1996 to 1999. Coupled with the uniform December 31 2006 censoring date, this led to large numbers of men with exactly 11, 10, 9 or 8 years of follow-up. Tracked backwards in time (i.e. from right to left), the population-time plot shows the recruitment pattern from its beginning in 1991, and the January 1 entries in successive years. Tracked forwards in time (i.e. from left to right), the plot for the first three years shows attrition due entirely to death (mainly from other causes). Since the Swedish and Belgian centres were the last to complete recruitment in December 2003, the minimum potential follow-up is three years. Tracked further forwards in time (i.e. after year 3) the attrition is a combination of deaths and staggered entries. As we can see, population-time plots summarise a wealth of information about the study into a simple graph.

Next, layers for the case series and base series are added. The y-axis location of each case moment is sampled at random vertically on the plot to avoid having all points along the upper edge of the gray area (Figure 1B). By randomly distributing the cases, we can get a sense of the incidence density. In Figure 1C, we see that more events are observed at later follow-up times. Therefore, a constant hazard model would not be appropriate in this instance as it would overestimate the incidence earlier on in time, and underestimate it later on. Finally, the base series is sampled uniformly from the study base (Figure 1D). The reader should refer to the package vignettes for more examples and a detailed description of how to modify the aesthetics of a population-time plot.

Parametric modeling

The parametric modeling step was separated into three parts:

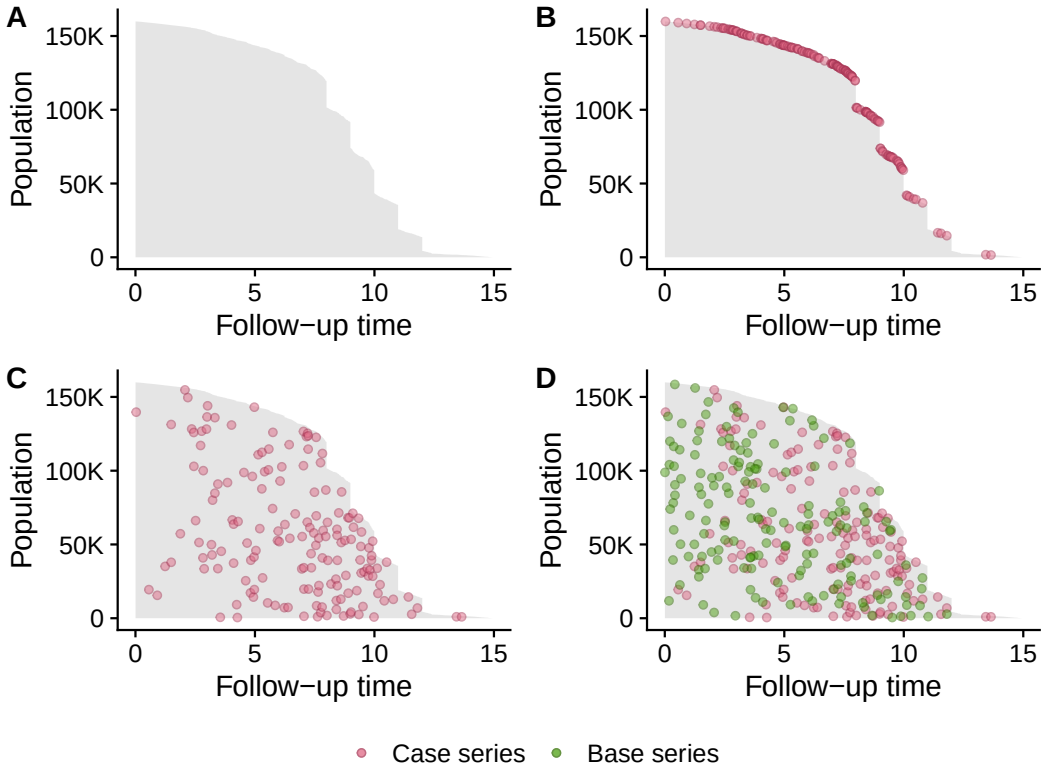


Figure 1: Population time plot for the ERSPC dataset. A: The gray area can be thought of as N=159,893 infinitely thin horizontal rectangles ordered by length of follow-up. B: The red points correspond to when death has occurred for any one of those infinitely thin rectangles. C: To improve visibility, these red points are randomly redistributed along their respective x-coordinates, providing a visualization of incidence density. More events are observed at later follow-up times, motivating the use of non-constant hazard models. D: The base series, a representative sample of the entire grey area, is represented by the green points.

1. case-base sampling;
2. estimation of the smooth hazard function;
3. estimation of the survival function.

By separating the sampling and estimation functions, we allow the possibility of users implementing more complex sampling scheme (as described in Saarela (2016)), or more complex study designs (e.g. time-varying exposure).

The sampling scheme selected for `sampleCaseBase` was described in Hanley & Miettinen (2009): we first sample along the "person" axis, proportional to each individual's total follow-up time, and then we sample a moment uniformly over their follow-up time. This sampling scheme is equivalent to the following picture: imagine representing the total follow-up time of all individuals in the study along a single dimension, where the follow-up time of the next individual would start exactly when the follow-up time of the previous individual ends. Then the base series could be sampled uniformly from this one-dimensional representation of the overall follow-up time. In any case, the output is a dataset of the same class as the input, where each row corresponds to a person-moment. The covariate profile for each such person-moment is retained, and an offset term is added to the dataset. This output could then be used to fit a smooth hazard function, or for visualization of the base series.

Next, the fitting function `fitSmoothHazard` starts by looking at the class of the dataset: if it was generated from `sampleCaseBase`, it automatically inherited the class `cbData`. If the dataset supplied to `fitSmoothHazard` does not inherit from `cbData`, then the fitting function starts by calling `sampleCaseBase` to generate the base series. In other words, users can bypass `sampleCaseBase` altogether and only worry about the fitting function `fitSmoothHazard`.

The fitting function retains the familiar formula interface of `glm`. The left-hand side of the formula should be the name of the column corresponding to the event type. The right-hand side can be any combination of the covariates, along with an explicit functional form for the time variable. Note that non-proportional hazard models can be achieved at this stage by adding an interaction term involving time (cf. Case Study 1 below). The offset term does not need to be specified by the user, as it

is automatically added to the formula before calling `glm`.

To fit the hazard function, we provide several approaches that are available via the `family` parameter. These approaches are:

- `glm`: This is the familiar logistic regression.
- `glmnet`: This option allows for variable selection using the elastic-net (Zou and Hastie 2005) penalty (cf. Case Study 3). This functionality is provided through the **glmnet** package (Friedman, Hastie, and Tibshirani 2010).
- `gam`: This option provides support for *Generalized Additive Models* via the **mgcv** package (Hastie and Tibshirani 1987).

In the case of multiple competing events, the hazard is fitted via multinomial regression as performed by the **VGAM** package. We selected this package for its ability to fit multinomial regression models with an offset.

Once a model-fit object has been returned by `fitSmoothHazard`, all the familiar summary and diagnostic functions are available: `print`, `summary`, `predict`, `plot`, etc. Our package provides one more functionality: it computes risk functions from the model fit. For the case of a single event, it uses the familiar identity

$$S(t) = \exp\left(-\int_0^t \lambda(u; X)du\right). \tag{6}$$

The integral is computed using either the numerical or Monte-Carlo integration. The risk function (or cumulative distribution function) is then defined as

$$F(t) = 1 - S(t). \tag{7}$$

For the case of a competing-event analysis, the event-specific risk is computed using the following procedure: first, we compute the overall survival function (i.e. for all event types):

$$S(t) = \exp\left(-\int_0^t \lambda(u; X)du\right), \qquad \lambda(t; X) = \sum_{j=1}^{J} \lambda_j(t; X).$$

From this, we can derive the cause-specific subdensities:

$$f_j(t) = \lambda_j(t)S(t).$$

By integrating these subdensities, we obtain the cause-specific CI functions:

$$CI_j(t) = \int_0^t f_j(u)du.$$

Again, the integrals are computed using either numerical integration (via the trapezoidal rule) or Monte Carlo integration. This option is controlled by the argument `method` of the `absoluteRisk` function.

Finally, the output from `absoluteRisk` can be passed to a method `confint` to compute confidence bands around the survival function, as described in the Theoretical Details section. These bands are only valid when `family = "glm"` as it relies on the asymptotic normality of the estimator. Currently, this is only available for the single-event setting.

5 Illustration of package

In this section, we illustrate the main functions of the **casebase** package through three case studies. Each one showcases a different type of analysis. First, we show how to model non-constant and non-proportional hazards through a flexible specification of time. Then we perform a competing risk analysis and compare our results with the Cox model and the Fine-Gray model. The third case study illustrates how to perform variable selection in high-dimensional datasets.

Case study 1—flexible modeling of the hazard function

For our first case study, we return to the ERSPC study and investigate the differences in risk between the control and screening arms. Previous re-analyses of these data suggest that the 20% reduction in prostate cancer death due to screening was an underestimate (Hanley 2010). The estimated 20% (from a proportional hazards model) did not account for the delay between screening and the time the effect is expected to be observed. As a result, the null effects in years 1–7 masked the substantial reductions

that began to appear from year 8 onward. This motivates the use of a time-dependent hazard ratio which can easily be fit with the **casebase** package by including an interaction term with time in the model. We fit a flexible hazard by using a smooth function of time modeled with a penalized cubic spline basis with 2 degrees of freedom (implemented in the `survival::pspline` function). The model is fit using `fitSmoothHazard` with the familiar formula interface:

```
fit <- fitSmoothHazard(DeadOfPrCa ~ pspline(Follow.Up.Time, df = 2) * ScrArm,
                       data = ERSPC, ratio = 10)
```

The output object from `fitSmoothHazard` inherits from the `singleEventCB` and `glm` classes. For this reason, we can leverage the `summary` method for `glm` objects to output a familiar summary of the results:

```
summary(fit)
#> Fitting smooth hazards with case-base sampling
#>
#> Sample size: 159893
#> Number of events: 540
#> Number of base moments: 5400
#> ----
#>
#> Call:
#> fitSmoothHazard(formula = DeadOfPrCa ~ pspline(Follow.Up.Time,
#>     df = 2) * ScrArm, data = ERSPC, ratio = 10)
#>
#> Deviance Residuals:
#>    Min      1Q  Median      3Q     Max
#> -1.168  -0.486  -0.414  -0.215   3.262
#>
#> Coefficients:
#>                                                    Estimate Std. Error
#> (Intercept)                                         -13.81       9.98
#> pspline(Follow.Up.Time, df = 2)1                      2.66      10.96
#> pspline(Follow.Up.Time, df = 2)2                      6.43       9.73
#> pspline(Follow.Up.Time, df = 2)3                      5.57      10.10
#> pspline(Follow.Up.Time, df = 2)4                      7.27       9.90
#> pspline(Follow.Up.Time, df = 2)5                      6.54      10.10
#> pspline(Follow.Up.Time, df = 2)6                     10.82      10.03
#> pspline(Follow.Up.Time, df = 2)7                    -11.74      30.13
#> ScrArmScreening group                                 9.22      13.35
#> pspline(Follow.Up.Time, df = 2)1:ScrArmScreening group   -9.25      14.85
#> pspline(Follow.Up.Time, df = 2)2:ScrArmScreening group   -9.80      12.97
#> pspline(Follow.Up.Time, df = 2)3:ScrArmScreening group   -9.04      13.54
#> pspline(Follow.Up.Time, df = 2)4:ScrArmScreening group   -9.39      13.23
#> pspline(Follow.Up.Time, df = 2)5:ScrArmScreening group  -10.65      13.55
#> pspline(Follow.Up.Time, df = 2)6:ScrArmScreening group   -8.86      13.46
#> pspline(Follow.Up.Time, df = 2)7:ScrArmScreening group  -11.58      36.92
#>                                                    z value Pr(>|z|)
#> (Intercept)                                         -1.38     0.17
#> pspline(Follow.Up.Time, df = 2)1                     0.24     0.81
#> pspline(Follow.Up.Time, df = 2)2                     0.66     0.51
#> pspline(Follow.Up.Time, df = 2)3                     0.55     0.58
#> pspline(Follow.Up.Time, df = 2)4                     0.73     0.46
#> pspline(Follow.Up.Time, df = 2)5                     0.65     0.52
#> pspline(Follow.Up.Time, df = 2)6                     1.08     0.28
#> pspline(Follow.Up.Time, df = 2)7                    -0.39     0.70
#> ScrArmScreening group                                0.69     0.49
#> pspline(Follow.Up.Time, df = 2)1:ScrArmScreening group  -0.62     0.53
#> pspline(Follow.Up.Time, df = 2)2:ScrArmScreening group  -0.76     0.45
#> pspline(Follow.Up.Time, df = 2)3:ScrArmScreening group  -0.67     0.50
#> pspline(Follow.Up.Time, df = 2)4:ScrArmScreening group  -0.71     0.48
#> pspline(Follow.Up.Time, df = 2)5:ScrArmScreening group  -0.79     0.43
#> pspline(Follow.Up.Time, df = 2)6:ScrArmScreening group  -0.66     0.51
#> pspline(Follow.Up.Time, df = 2)7:ScrArmScreening group  -0.31     0.75
```

```
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 3619.1  on 5939  degrees of freedom
#> Residual deviance: 3359.1  on 5924  degrees of freedom
#> AIC: 3391
#>
#> Number of Fisher Scoring iterations: 7
```

As noted in the Theoretical Details section, the usual asymptotic results hold for likelihood ratio tests built using case-base sampling models. Therefore, we can easily test the significance of the spline term and its interaction with time:

```
anova(fit, test = "LRT")
#> Analysis of Deviance Table
#>
#> Model: binomial, link: logit
#>
#> Response: DeadOfPrCa
#>
#> Terms added sequentially (first to last)
#>
#>
#>                                       Df Deviance Resid. Df Resid. Dev
#> NULL                                                  5939       3619
#> pspline(Follow.Up.Time, df = 2)        7    246.6      5932       3373
#> ScrArm                                 1      5.6      5931       3367
#> pspline(Follow.Up.Time, df = 2):ScrArm 7      7.9      5924       3359
#>                                       Pr(>Chi)
#> NULL
#> pspline(Follow.Up.Time, df = 2)          <2e-16 ***
#> ScrArm                                    0.018 *
#> pspline(Follow.Up.Time, df = 2):ScrArm    0.343
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Similarly, to compare different models (e.g. time modeled linearly), we could compute Akaike's Information Criterion (AIC) for each model.

Time-dependent hazard ratios

In what follows, the hazard ratio for a variable X is defined as

$$\frac{\lambda\left(t|X=x_1,\mathbf{Z}=\mathbf{z};\widehat{\beta}\right)}{\lambda(t|X=x_0,\mathbf{Z}=\mathbf{z};\widehat{\beta})}$$

where $\lambda(t|\cdot;\widehat{\beta})$ is the hazard rate as a function of the variable t (which is usually time, but can be any other continuous variable), x_1 is the value of X for the exposed group, x_0 is the value of X for the unexposed group, $\mathbf{Z}$ are other covariates in the model which are equal to $\mathbf{z}$ in both the exposed and unexposed group, and $\widehat{\beta}$ are the estimated regression coefficients. As indicated by the formula above, it is most instructive to plot the hazard ratio as a function of a variable t only if there is an interaction between t and X. Otherwise, the resulting plot will simply be a horizontal line across time.

The `plot` method with `type="hr"` for objects of class `singleEventCB` can be used to compute time-dependent hazard ratios and confidence intervals. In Figure 2, we show the estimated hazard ratio and 95% confidence interval for screening vs. control group as a function of time. Note that we must specify the covariate profile for the reference group and times for the predicted hazards.

```
new_time <- seq(1, 12, by  = 0.1)
new_data <- data.frame(ScrArm = factor("Control group",
                                       levels = c("Control group","Screening group")),
                       Follow.Up.Time = new_time)
plot(fit, type = "hr", newdata = new_data,
     var = "ScrArm", xvar = "Follow.Up.Time", ci = TRUE)
```

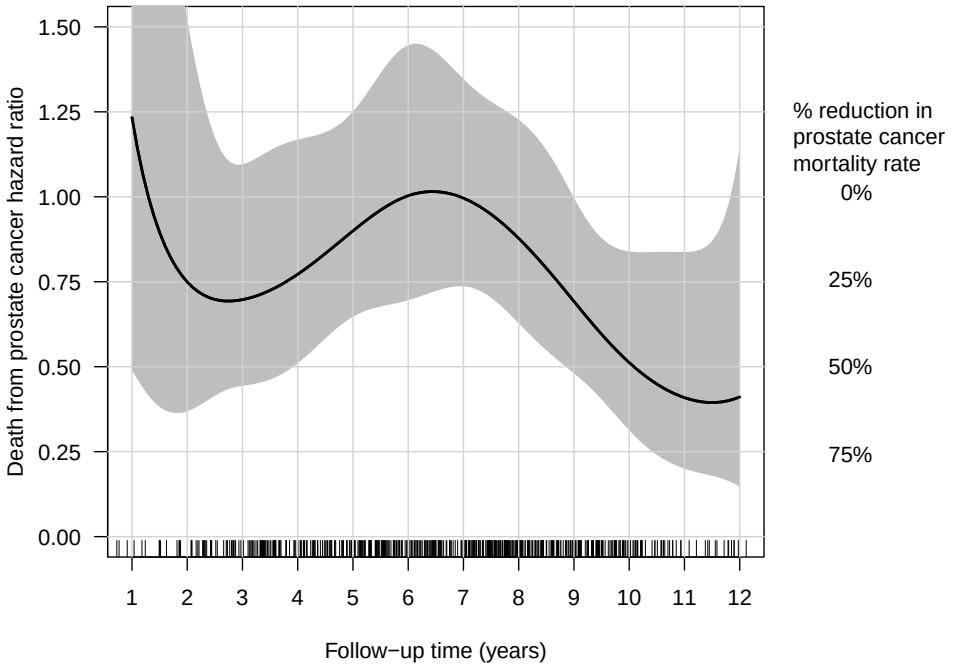


Figure 2: Estimated hazard ratio and 95% confidence interval for screening vs. control group as a function of time in the ERSPC dataset. Hazard ratios are estimated from fitting a parametric hazard model as a function of the interaction between a cubic pspline basis (df=2) of follow-up time and treatment arm. 95% confidence intervals are calculated using the delta method. The plot shows that the effect of screening only begins to become statistically apparent by year 7. The 25-60% reductions seen in years 8-12 of the study suggests a much higher reduction in prostate cancer due to screening than the single overall 20% reported in the original article.

The plot shows that the effect of screening only becomes statistically apparent by year 7 and later. The 25-60% reductions seen in years 8-12 of the study suggests a much higher reduction in prostate cancer due to screening than the single overall 20% reported in the original article.

A more parsimonious model based on these results could be constructed as follows:

$$\log \lambda\left(t \mid \mathbf{Z}\right) = \begin{cases} \alpha, & t < t_0 \\ \alpha + \beta Z(t - t_0), & t \geq t_0 \end{cases},$$

where $t_0 \approx 7$ years. This model corresponds to a hazard ratio that is constant and equal to 1 until $t = t_0$, after which it decreases exponentially. If we fix the value t_0, we can easily implement this model using our package; for example:

```
fitSmoothHazard(DeadOfPrCa ~ as.integer(Follow.Up.Time >= t0) :
                    ScrArm : I(Follow.Up.Time - t0),
                data = ERSPC)
```

Here, we use the binary variable `as.integer(Follow.Up.Time >= t0)` in order to write the two cases of our formula above in a more compact way. We also use the function `I()`, which allows us to specify new variables within the formula using normal `R` code. This is required, as otherwise the symbol `-` would be interpreted as a formula operator. Finally, the term `ScrArm : I(Follow.Up.Time - t0)` represents the product $Z(t - t_0)$ in the equation above. We use the operator `:` instead of `*` in order to only include the interaction term, not the main effects.

Alternatively, the breakpoint t_0 could also be estimated by combining case-base sampling with segmented regression. However, this extension is beyond the current scope of `casebase`.

Hazard functions

Modeling the hazard function directly allows us to easily visualize it with the `plot` method and `type="hazard"` for objects of class `singleEventCB`. We plot the hazard functions for both treatment arms in Figure 3. The pattern we see is consistent with the population-time plot shown in Figure 1C, where more events are observed at later follow-up times. The drop at the end can be explained by the fact that very few observations were followed for the entire 15 year period.

```
plot(fit, type = "hazard",
     hazard.params = list(xvar = "Follow.Up.Time",
                          by = "ScrArm"))
#> Conditions used in construction of plot
#> ScrArm: Control group / Screening group
#> offset: 0
```

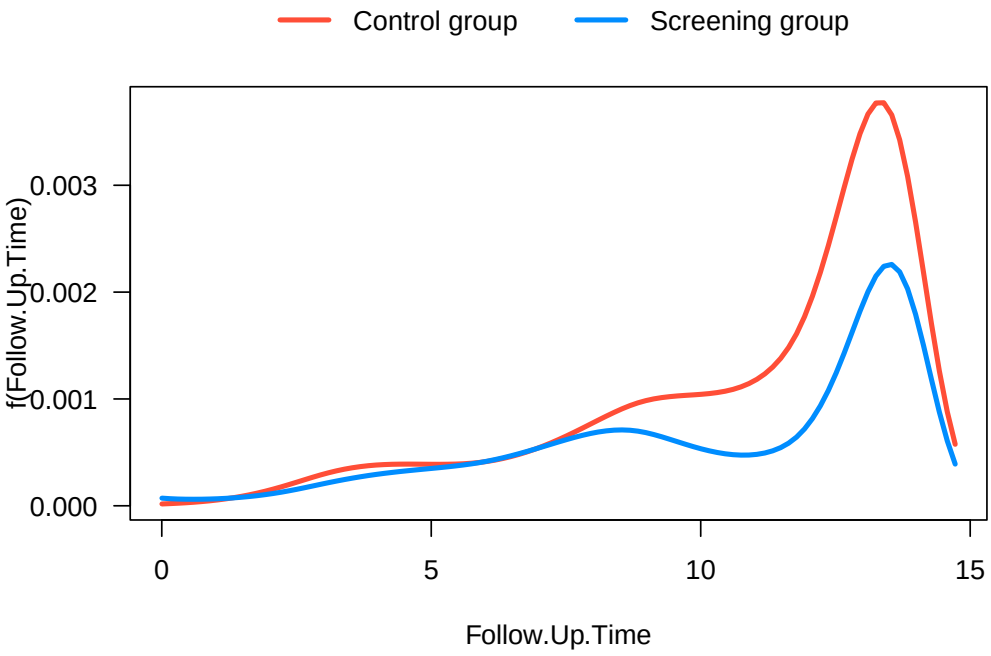


Figure 3: Estimated hazard functions for control and screening groups in the ERSPC dataset. Hazards are estimated from fitting a parametric model with casebase sampling as a function of the interaction between a cubic pspline basis (df=2) of follow-up time and treatment arm. The package vignettes provide a detailed description of how to plot hazard functions for any combination of covariates along with confidence bands.

Absolute risk

Next, the `absoluteRisk` function takes as input the `singleEventCB` object and returns a matrix where each column corresponds to the covariate profiles specified in the `newdata` argument, and each row corresponds to time points specified by the `time` argument:

```
new_data <- data.frame(ScrArm = c("Control group", "Screening group"))
new_time <- seq(0,14,0.1)
risk <- absoluteRisk(fit, time = new_time, newdata = new_data)
```

We can subsequently compute confidence intervals for the risk function using the method `confint.absRiskCB`:

```
conf_ints <- confint(risk, fit)
```

```
head(conf_ints)
#>   time estimate conf.low conf.high      cov_prof
#> 1  0.0  0.0e+00  0.0e+00   0.0e+00 Control group
#> 2  0.1  1.8e-06  1.9e-07   1.2e-05 Control group
#> 3  0.2  3.9e-06  4.8e-07   2.2e-05 Control group
#> 4  0.3  6.1e-06  9.0e-07   3.2e-05 Control group
#> 5  0.4  8.7e-06  1.5e-06   4.1e-05 Control group
#> 6  0.5  1.2e-05  2.3e-06   5.0e-05 Control group
```

In Figure 4, we see the 95% confidence bands around the estimates. We also overlay the Kaplan-Meier curves as a reference.

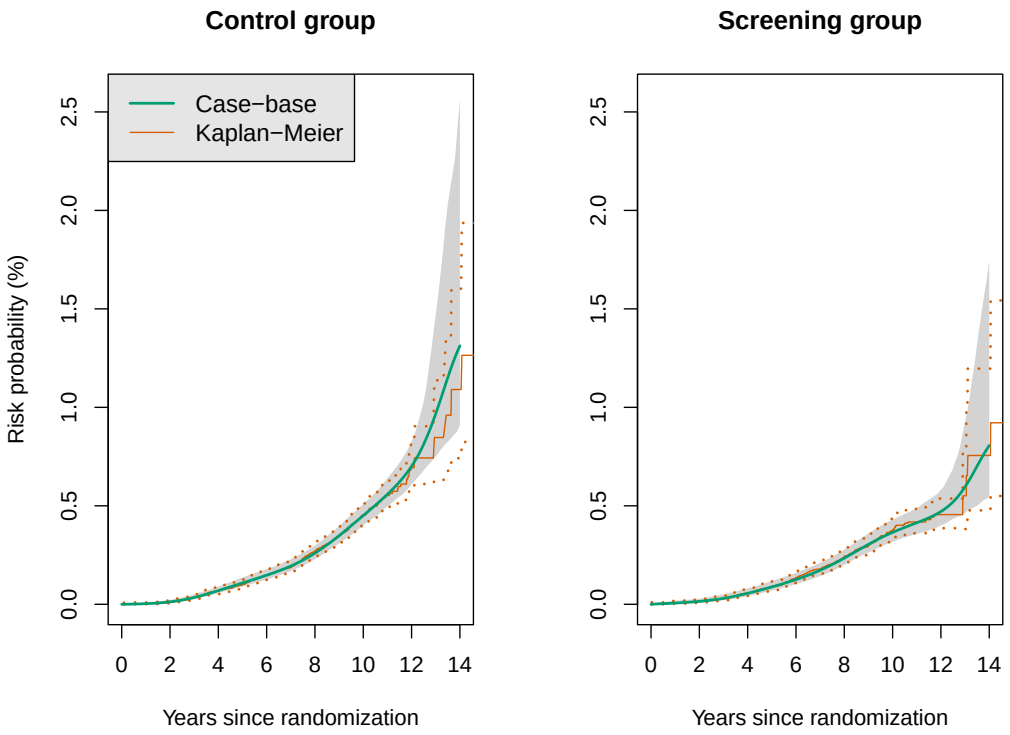


Figure 4: Risk function estimates for control and screening groups in the ERSPC data using case-base sampling and Kaplan-Meier, along with 95% confidence bands. The smooth curve (case-base sampling) vs. step function (Cox model) highlight one of the main differences between the two approaches. The larger confidence bands in the later years is due to the relatively few number of individuals who were followed for more than 12 years.

Case study 2—competing risk analysis

In this case study, we show how case-base sampling can be used in the context of a competing risk analysis. For illustrative purposes, we use the same data that was used in Scrucca *et al* (2010). The data was downloaded from the first author's website, and it is now available as part of the **casebase** package.

The data contains information on 177 patients who received a stem-cell transplant for acute leukemia. The event of interest is relapse, but other competing causes (e.g. death, progression, graft failure, graft-versus-host disease) were also recorded. Several covariates were captured at baseline: sex, disease type (acute lymphoblastic or myeloblastic leukemia, abbreviated as ALL and AML, respectively), disease phase at transplant (Relapse, CR1, CR2, CR3), source of stem cells (bone marrow and peripheral blood, coded as BM+PB, or only peripheral blood, coded as PB), and age.

First, we look at a population-time plot to visualize the incidence density of both relapse and the competing events. In Figure 5, failure times are highlighted on the plot using red dots for the event of interest and blue dots for competing events. In this plot, we see evidence of a non-constant hazard function: the density of points is larger at the beginning of follow-up than at the end.

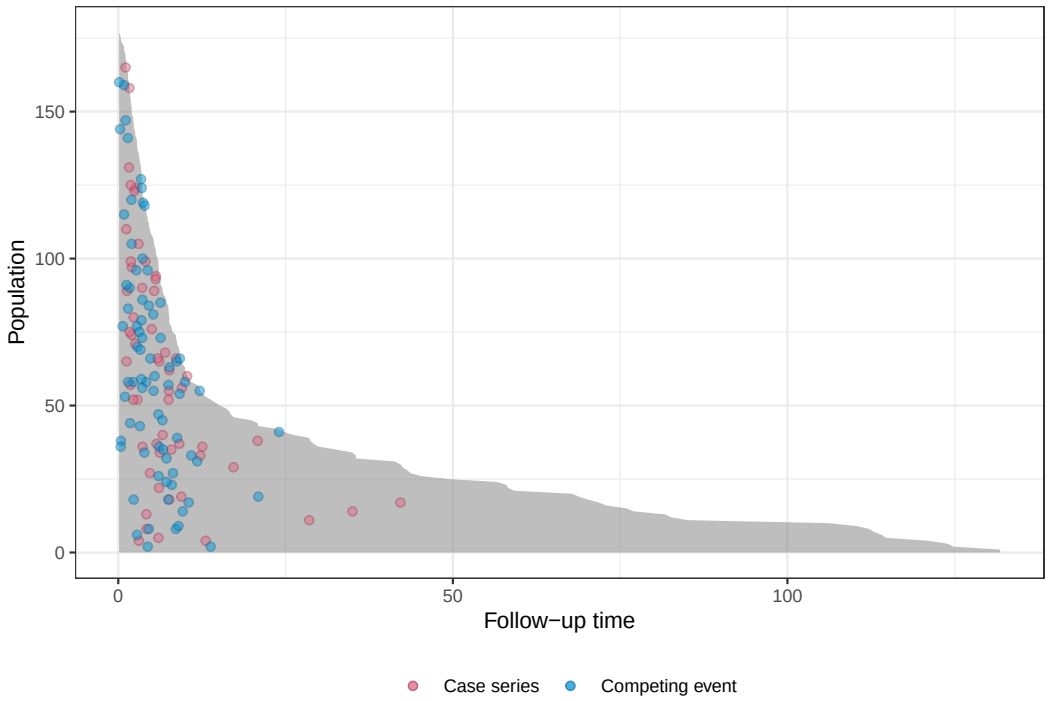


Figure 5: Population-time plot for the stem-cell transplant study with both relapse and competing events. The area representing the population time is shown in gray, with subjects having the least amount of observation time plotted at the top of the y-axis. The y-axis location of each case series and competing event moment is sampled at random vertically on the plot to avoid having all points along the upper edge of the gray area. The density of points at the beginning of follow-up relative to the end indicates a non-constant hazard function.

Our main objective is to compute the cumulative incidence of relapse for a given set of covariates. We start by fitting a smooth hazard to the data using a linear term for time:

```
model_cb <- fitSmoothHazard(
  Status ~ ftime + Sex + D + Phase + Source + Age,
  data = bmtcrr,
  ratio = 100,
  time = "ftime"
)
```

We want to compare our hazard ratio estimates to that obtained from a Cox regression (using the **survival** package version 3.2-13).

```
library(survival)
# Prepare data for coxph
bmtcrr_cox <- transform(bmtcrr,
                        id = seq_len(nrow(bmtcrr)),
                        Status = factor(Status))

model_cox <- coxph(Surv(ftime, Status) ~ Sex + D + Phase + Source + Age,
                   data = bmtcrr_cox, id = id)
```

From the fit object, we can extract both the hazard ratios and their corresponding confidence intervals. These quantities appear in Table 2. As we can see, the type of disease corresponds to a significant hazard ratio: the hazard for AML is about half that for ALL. Moreover, being in relapse at transplant is associated with a hazard ratio of 4.22 when compared to CR1.

Given the estimate of the hazard functions obtained using case-base sampling, we can compute the cumulative incidence curve for a fixed covariate profile. We perform this computation for a 35 year old woman who received a stem-cell transplant from peripheral blood at relapse. We compare the absolute risk curve for such a woman with ALL with that for a similar woman with AML.

Table 2: Estimates and confidence intervals for the hazard ratios for each coefficient. Both estimates from case-base sampling and Cox regression are presented.

	Case-Base		Cox	
Covariates	HR	95% Conf.	HR	95% Conf.
Sex	0.73	(0.42, 1.27)	0.68	(0.39, 1.21)
Disease	0.54	(0.3, 0.98)	0.52	(0.28, 0.94)
Phase (CR2 vs. CR1)	1.19	(0.48, 2.97)	1.21	(0.47, 3.09)
Phase (CR3 vs. CR1)	1.44	(0.37, 5.64)	1.67	(0.46, 6.08)
Phase (Relapse vs. CR1)	4.22	(1.95, 9.12)	4.55	(1.98, 10.46)
Source	1.46	(0.48, 4.46)	1.46	(0.47, 4.53)
Age	0.99	(0.97, 1.02)	0.99	(0.97, 1.02)

Next, we compare our estimates to that obtained from a corresponding Fine-Gray model (1999). The Fine-Gray model is a semiparametric model for the cause-specific *subdistribution hazard*, i.e. the function $d_j(t)$ such that

$$CI_j(t) = 1 - \exp\left(-\int_0^t d_j(u)\mathrm{d}u\right),$$

where $CI_j(t)$ is the cause-specific CI. The Fine-Gray model allows to directly assess the effect of a covariate on the subdistribution hazard, as opposed to the cause-specific hazard. For the computation, we use the **timereg** package (Thomas H. Scheike and Zhang 2011):

```
library(timereg)
model_fg <- comp.risk(Event(ftime, Status) ~ const(Sex) + const(D) +
                        const(Phase) + const(Source) + const(Age),
                      data = bmtcrr, cause = 1, model = "fg")

# Estimate CI curve
risk_fg <- predict(model_fg, newdata, times = time_points)
```

We can also estimate the CI for relapse using the Cox model and the Aalen-Johansen estimator:

```
# Estimate absolute risk curve
risk_cox <- survfit(model_cox, newdata = newdata)
```

Figure 6 shows the CI curves for all three models. As we can see, all three approaches agree quite well for AML; however, for ALL, there seems to be a difference of about 5% between the Fine-Gray curve and the curves estimated using case-base sampling and Cox regression. This difference does not appear to be significant: the curve from case-base sampling is contained within a 95% confidence band around the Fine-Gray absolute risk curve (figure not shown).

Case study 3—variable selection

For the third case study, we show how **casebase** can also be used for variable selection through regularized estimation of the hazard function as given by Equation (5). We note that this is different than the semiparametric model Coxnet, which regularizes the Cox partial likelihood. To illustrate this functionality, we use the dataset from the Study to Understand Prognoses Preferences Outcomes and Risks of Treatment (SUPPORT) (Knaus et al. 1995).[3] The SUPPORT dataset tracks death in five American hospitals within individuals who are considered seriously ill. The cleaned and imputed data consists of 9104 observations and 30 variables, and it is available as part of the **casebase** package. In the comparisons below, all covariates except `sps` and `aps` were used. These two variables correspond to scores for predicting the outcome that were developed as part of the original study. For more information about this dataset, the reader is encouraged to look at the documentation in our package.

For our penalized case-base model, we opt for the natural log of time which corresponds to a Weibull distribution. For fitting the penalized hazard, we use `fitSmoothHazard.fit`, which is a matrix interface to the `fitSmoothHazard` function. The `fitSmoothHazard` and `fitSmoothHazard.fit`

[3]The original data is available online from the Department of Biostatistics at Vanderbilt University: https://biostat.app.vumc.org/wiki/Main/SupportDesc

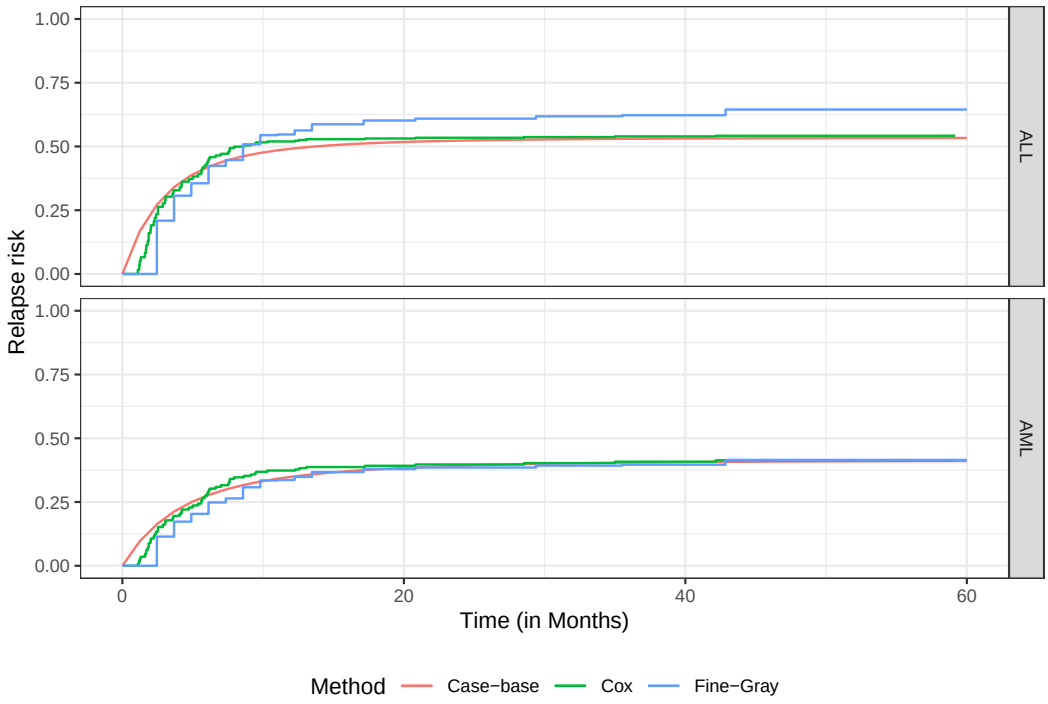


Figure 6: Cumulative Incidence curve for a fixed covariate profile and the two disease groups. The estimate obtained from case-base sampling is compared to the Fine-Gray and Aalen-Johansen estimates. In general, the three approaches agree quite well for AML, while there seems to be a difference of about 5% between the Fine-Gray curve and the curves estimated using case-base sampling and Cox regression for ALL. However, this difference does not appear to be significant as the curve from case-base sampling is contained within a 95% confidence band around the Fine-Gray absolute risk curve (figure not shown).

functions sample the case and base series, calculate the required offset, and transform the data to match the expected input of the **glmnet** package. The penalized logistic regression is then fit for multiple values of the tuning parameter using the function `glmnet::cv.glmnet` and the binomial family. To `fitSmoothHazard.fit`, we supply both a matrix `y` containing the time and event variables, and a matrix `x` containing all other covariates. We apply the lasso penalty by setting `alpha = 1` and assign a `penalty.factor` (w_j; cf. Equation (5)) of 0 to the time variable to ensure it is in the selected model. We compare our approach to both Cox regression, and lasso penalized Cox regression (fitted via the **glmnet** package and using the Cox family).

To compare the performance of our models, we split the data into 95% training and 5% test sets. To assess both discrimination and calibration, we use a time-dependent version of the classical Brier score that is adjusted for censoring (Graf et al. 1999). The Brier score can be used with both parametric and semi-parametric models. We use the **riskRegression** package to compute these scores for all models.

```
# Create matrices for inputs
x <- model.matrix(death ~ . - d.time - aps - sps,
                  data = train)[, -c(1)] # Remove intercept
y <- data.matrix(subset(train, select = c(d.time, death)))

# Regularized logistic regression to estimate hazard
pen_cb <- casebase::fitSmoothHazard.fit(x, y,
  family = "glmnet",
  time = "d.time", event = "death",
  formula_time = ~ log(d.time), alpha = 1,
  ratio = 10, standardize = TRUE,
  penalty.factor = c(0, rep(1, ncol(x)))
)
```

In Figure 7, we show the coefficient estimates for covariates that we selected by both penalized Cox and penalized case-base. We note that both penalized approaches produce similar results. We can

also clearly see the shrinkage effect owing to the ℓ_1 penalty.

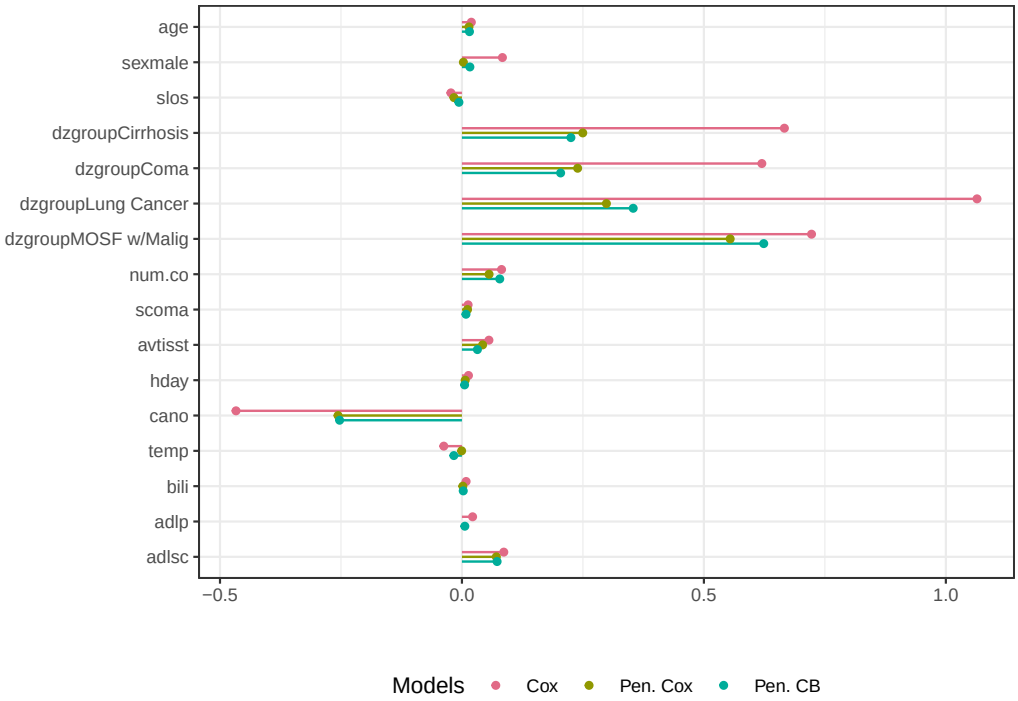


Figure 7: Coefficient estimates from the Cox model (Cox), penalized Cox model using the glmnet package (Pen. Cox), and our approach using penalized case-base sampling (Pen. CB). Only the covariates that were selected by both penalized approaches are shown. The shrinkage of the coefficient estimates for Pen. Cox and Pen. CB occurs due to the ℓ_1 penalty.

We then compare the risk estimation over the test set. The predicted probabilities for each test set observation are averaged, resulting in the absolute risk curves shown in Figure 8A. The Kaplan-Meier curve is calculated on the test set only. We see minimal differences between the four approaches across follow-up-time. Note that the apparent smoothness of the Cox and penalized Cox curves is due to the large number of observations in the training set, which is used to derive the Breslow estimate of the baseline hazard. As described above, we compare the performance between the models by computing the Brier scores over time. In Figure 8B, we see that the adjusted models all perform similarly, outperforming the Kaplan-Meier estimate.

6 Discussion

In this article, we presented the R package **casebase**, which provides functions for fitting smooth parametric hazards and estimating risk functions using case-base sampling. Our package also provides several functions to produce graphical summaries of the data and the results. We outlined the theoretical underpinnings of the approach, we provided details about our implementation, and we illustrated the merits of the case-base framework and the package through three case studies.

As a methodological framework, case-base sampling is very flexible. Some of this flexibility has been explored before in the literature: for example, Saarela and Hanley (2015) used case-base sampling to model a time-dependent exposure variable in a vaccine safety study. As another example, Saarela and Arjas (2015) combined case-base sampling and a Bayesian non-parametric framework to compute individualized risk assessments for chronic diseases. In the case studies above, we further explored this flexibility along two fronts. On the one hand, we showed how splines could be used as part of the linear predictor to model the effect of time on the hazard. This strategy yielded estimates of the survival function that were qualitatively similar to semiparametric estimates derived from Cox regression; however, case-base sampling led to estimates of the survival function that *vary smoothly in time* and are thus easier to interpret. On the other hand, we also displayed the flexibility of case-base sampling by showing how it could be combined with penalized logistic regression to perform variable selection. Furthermore, the second case study showed how case-base sampling can be applied to competing risks settings. It should be noted that the three case studies presented above only considered covariates that were fixed at baseline. In one of the package vignettes, we use the Stanford Heart

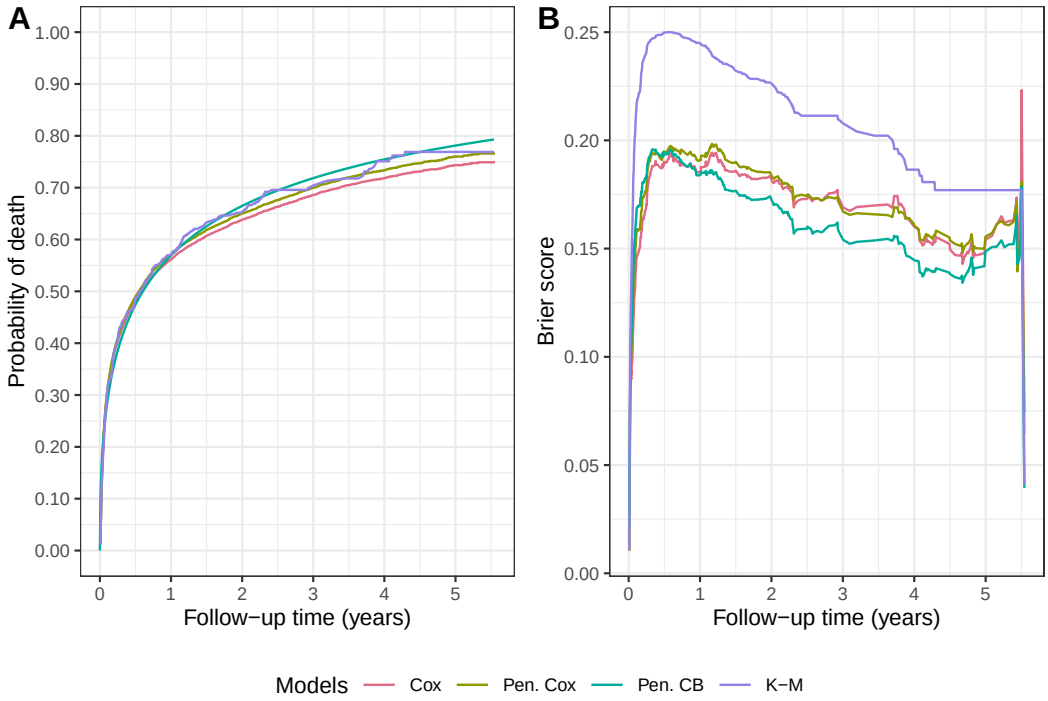


Figure 8: Comparison of Cox regression (Cox), penalized Cox regression (Pen. Cox), penalized case-base sampling estimation (Pen. CB), and Kaplan-Meier (K-M). (A) Probability of death as a function of follow-up time which is the average of the predicted probabilities for each test set observation. The Kaplan-Meier curve is calculated on the test set only. We see minimal differences between the four approaches across follow-up-time for the absolute risk curves. Note that the apparent smoothness of the Cox and penalized Cox curves is due to the large number of observations in the training set, which is used to derive the Breslow estimate of the baseline hazard. (B) Brier score as a function of follow-up time, where a lower score corresponds to better performance. We see that the adjusted models all perform similarly, outperforming the Kaplan-Meier estimate.

Transplant data Crowley and Hu (1977) to show how case-base sampling can also be used in the context of time-dependent exposure. In this study, the exposure period was defined as the week following vaccination. Hence, the main covariate of interest, i.e. exposure to the vaccine, was changing over time. In this context, case-base sampling offers an efficient alternative to nested case-control designs or self-matching.

Even though we did not illustrate it in this article, case-base sampling can also be combined with the framework of *generalized additive models*. This functionality has already been implemented in the package. Similarly, case-base sampling can be combined with quasi-likelihood estimation to fit survival models that can account for the presence of over-dispersion. All of these examples illustrate how the case-base sampling framework in general, and the package **casebase** in particular, allows the user to fit a broad and flexible family of survival functions.

Poisson regression can also be used to estimate the full hazard by discretizing time. However, this method requires user input on the number of intervals, or equivalently, on the cut points. This choice made by the user can have a significant impact on the model. Small intervals may result in many empty, non-informative bins. This may cause convergence issues for the Newton-Raphson procedure (Kalbfleisch and Prentice 2011). If the intervals are too wide, the nonlinear trends that are present in the hazard may be masked. Rather than discretizing time like in Poisson regression, case-base sampling provides a continuous-time approach to using GLMs for estimating hazard functions.

As presented in Hanley & Miettinen (2009), case-base sampling is comprised of three steps: 1) sampling a case series and a base series from the study; 2) fit the log-hazard as a linear function of predictors (including time); and 3) use the fitted hazard to estimate the risk function. Accordingly, our package provides functions for each step. Moreover, the simple interface of the `fitSmoothHazard` function resembles the `glm` interface. This interface should look familiar to new users. Our modular approach also provides a convenient way to extend our package for new sampling or fitting strategies.

In the case studies above, we compared the performance of case-base sampling with that of Cox regression and Fine-Gray models. In terms of function interface, **casebase** uses a formula interface that

is closer to that of `glm`, in that the event variable is the only variable appearing on the left-hand side of the formula. By contrast, both `survival::coxph` and `timereg::comp.risk` use arrays that capture both the event type and time. Both approaches to modeling yield user-friendly code. However, in terms of output, both approaches differ significantly. Case-base sampling produces smooth hazards and smooth survival curves, whereas Cox regression and Fine-Gray models produce stepwise CIs and never explicitly model the hazard function. Qualitatively, we showed that by using splines in the linear predictor, all three models yielded similar curves.

Our choice of modeling the log-hazard as a linear function of covariates allows us to develop a simple computational scheme for estimation. However, as a downside, it does not allow us to model location and scale parameters separately like the package **flexsurv**. For example, if we look at the Weibull distribution as parametrised in `stats::pweibull`, the log-hazard function is given by

$$\log \lambda(t; \alpha, \beta) = [\log(\alpha/\beta) - (\alpha - 1)\log(\beta)] + (\alpha - 1)\log t,$$

where α, β are shape and scale parameters, respectively. Unlike **casebase**, the approach taken by **flexsurv** also allows the user to model the scale parameter as a function of covariates. Of course, this added flexibility comes at the cost of interpretability: by modeling the log-hazard directly, the parameter estimates from **casebase** can be interpreted as estimates of log-hazard ratios. To improve the flexibility of **casebase** at capturing the scale of a parametric family, we could replace the logistic regression with its quasi-likelihood counterpart and therefore model over- and under-dispersion with respect to the logistic likelihood. We defer the study of the properties and performance of such a model to a future article.

Future work will look at some of the methodological extensions of case-base sampling. First, to assess the quality of the model fit, we would like to study the properties of the residuals (e.g. Cox-Snell, martingale). More work needs to be done to understand these residuals in the context of the partial likelihood underlying case-base sampling. The resulting diagnostic tools could then be integrated in this package. Also, we are interested in extending case-base sampling to account for interval censoring. This type of censoring is very common in longitudinal studies, and many packages (e.g. **SmoothHazard**, **survival** and **rstpm2**) provide functions to account for it. Again, we hope to include any resulting methodology as part of this package.

In future versions of the package, we also want to increase the complement of diagnostic and inferential tools that are currently available. For example, we would like to include more functions to compute calibration and discrimination statistics (e.g. AUC) for our models. Saarela and Arjas (2015) also describe how to obtain a posterior distribution for the AUC from their model. Their approach could potentially be included in **casebase**. Finally, we want to provide more flexibility in how the case-base sampling is performed. This could be achieved by adding a `hazard` argument to the function `sampleCaseBase`. In this way, users could specify their own sampling mechanism. For example, they could provide a hazard that gives sampling probabilities that are proportional to the cardiovascular disease event rate given by the Framingham score (Saarela and Arjas 2015).

In conclusion, we presented the R package **casebase** which implements case-base sampling for fitting parametric survival models and for estimating smooth survival functions using the framework of generalized linear models. We strongly believe that its flexibility and its foundation on the familiar logistic regression model will make it appealing to new and established practitioners. The **casebase** package is freely available from the Comprehensive R Archive Network at https://cran.r-project.org/package=casebase. Interested users can visit http://sahirbhatnagar.com/casebase/ for detailed package documentation and vignettes.

7 Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and criticisms. We would also like to thank Yi Yang for helpful discussions on penalized regression models. Bhatnagar (RGPIN-2020-05133) and Turgeon (RGPIN-2021-04073) both gratefully acknowledge funding via a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN.

References

Aalen, Odd, Ornulf Borgan, and Hakon Gjessing. 2008. *Survival and Event History Analysis: A Process Point of View*. Springer Science & Business Media.

Allignol, Arthur, and Aurelien Latouche. 2019. "CRAN Task View: Survival Analysis." *CRAN Task View: Survival Analysis*. https://cran.r-project.org/web/views/Survival.html.

Arjas, Elja, and Pentti Haara. 1987. "A Logistic Regression Model for Hazard: Asymptotic Results." *Scandinavian Journal of Statistics*, 1–18.

Bhatnagar, Sahir, Maxime Turgeon, Jesse Islam, Olli Saarela, and James Hanley. 2021. *Casebase: Fitting Flexible Smooth-in-Time Hazards and Risk Functions via Logistic and Multinomial Regression*. https://CRAN.R-project.org/package=casebase.

Breslow, NE. 1972. "Discussion of the Paper by Dr Cox Cited Below." *Journal of the Royal Statistical Society: Series B (Methodological)* 34: 187–220.

Carstensen, Bendix. 2019. "Who Needs the Cox Model Anyway." http://bendixcarstensen.com/WntCma.pdf.

Clark, David A, Edward B Stinson, Randall B Griepp, John S Schroeder, Norman E Shumway, and Donald Harrison. 1971. "Cardiac Transplantation in Man." *Annals of Internal Medicine* 75 (1): 15–21.

Clements, Mark, and Xing-Rong Liu. 2019. *Rstpm2: Smooth Survival Models, Including Generalized Survival Models*. https://CRAN.R-project.org/package=rstpm2.

Crowley, John, and Marie Hu. 1977. "Covariance Analysis of Heart Transplant Survival Data." *Journal of the American Statistical Association* 72 (357): 27–36.

Efron, Bradley. 1988. "Logistic Regression, Survival Analysis, and the Kaplan-Meier Curve." *Journal of the American Statistical Association* 83 (402): 414–25.

Efron, Bradley, and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

Fine, Jason P, and Robert J Gray. 1999. "A Proportional Hazards Model for the Subdistribution of a Competing Risk." *Journal of the American Statistical Association* 94 (446): 496–509.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1). https://doi.org/10.18637/jss.v033.i01.

Gerds, Thomas Alexander, and Brice Ozenne. 2020. *riskRegression: Risk Regression Models and Prediction Scores for Survival Analysis with Competing Risks*. https://CRAN.R-project.org/package=riskRegression.

Goeman, J. J. 2010. "L1 Penalized Estimation in the Cox Proportional Hazards Model." *Biometrical Journal*, no. 52: –14.

Graf, Erika, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. 1999. "Assessment and Comparison of Prognostic Classification Schemes for Survival Data." *Statistics in Medicine* 18 (17-18): 2529–45.

Hanley, James A. 2010. "Mortality Reductions Produced by Sustained Prostate Cancer Screening Have Been Underestimated." *Journal of Medical Screening* 17 (3): 147–51.

Hanley, James A, and Olli S Miettinen. 2009. "Fitting smooth-in-time Prognostic Risk Functions via Logistic Regression." *The International Journal of Biostatistics* 5 (1).

Hastie, Trevor, and Robert Tibshirani. 1987. "Generalized Additive Models: Some Applications." *Journal of the American Statistical Association* 82 (398): 371–86.

Jackson, Christopher. 2016. "flexsurv: A Platform for Parametric Survival Modeling in R." *Journal of Statistical Software* 70 (8): 1–33. https://doi.org/10.18637/jss.v070.i08.

Kalbfleisch, John D, and Ross L Prentice. 2011. *The Statistical Analysis of Failure Time Data*. Vol. 360. John Wiley & Sons.

Knaus, William A, Frank E Harrell, Joanne Lynn, Lee Goldman, Russell S Phillips, Alfred F Connors, Neal V Dawson, et al. 1995. "The SUPPORT Prognostic Model: Objective Estimates of Survival for Seriously Ill Hospitalized Adults." *Annals of Internal Medicine* 122 (3): 191–203.

Liu, Zhihui, Benjamin Rich, and James A Hanley. 2014. "Recovering the Raw Data Behind a Non-Parametric Survival Curve." *Systematic Reviews* 3 (1): 151.

Mahani, Alireza, and Mansour Sharabiani. 2019. "Bayesian, and Non-Bayesian, Cause-Specific Competing-Risk Analysis for Parametric and Nonparametric Survival Functions: The R Package CFC." *Journal of Statistical Software* 89 (9): 1–29. https://doi.org/10.18637/jss.v089.i09.

Park, Mee Young, and Trevor Hastie. 2018. *Glmpath: L1 Regularization Path for Generalized Linear Models and Cox Proportional Hazards Model*. https://CRAN.R-project.org/package=glmpath.

Reid, Nancy. 1994. "A Conversation with Sir David Cox." *Statistical Science* 9 (3): 439–55.

Saarela, Olli. 2016. "A Case-Base Sampling Method for Estimating Recurrent Event Intensities." *Lifetime Data Analysis* 22 (4): 589–605.

Saarela, Olli, and Elja Arjas. 2015. "Non-Parametric Bayesian Hazard Regression for Chronic Disease Risk Assessment." *Scandinavian Journal of Statistics* 42 (2): 609–26.

Saarela, Olli, and James A Hanley. 2015. "Case-Base Methods for Studying Vaccination Safety." *Biometrics* 71 (1): 42–52.

Scheike, Thomas H, Klaus K Holst, and Jacob B Hjelmborg. 2014. "Estimating Twin Concordance for Bivariate Competing Risks Twin Data." *Statistics in Medicine* 33 (7): 1193–1204.

Scheike, Thomas H., and Mei-Jie Zhang. 2011. "Analyzing Competing Risk Data Using the R timereg Package." *Journal of Statistical Software* 38 (2): 1–15.

Schröder, Fritz H, Jonas Hugosson, Monique J Roobol, Teuvo LJ Tammela, Stefano Ciatto, Vera Nelen, Maciej Kwiatkowski, et al. 2009. "Screening and Prostate-Cancer Mortality in a Randomized

European Study." *New England Journal of Medicine* 360 (13): 1320–28.
Scrucca, L, A Santucci, and F Aversa. 2010. "Regression Modeling of Competing Risk Using R: An in Depth Guide for Clinicians." *Bone Marrow Transplantation* 45 (9): 1388.
Simon, Noah, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2011. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software* 39 (5): 1–13.
Therneau, Terry M. 2015. *A Package for Survival Analysis in s*. https://CRAN.R-project.org/package=survival.
Touraine, Célia, Thomas A. Gerds, and Pierre Joly. 2017. "SmoothHazard: An R Package for Fitting Regression Models to Interval-Censored Observations of Illness-Death Models." *Journal of Statistical Software* 79 (7): 1–22. https://doi.org/10.18637/jss.v079.i07.
Van der Vaart, Aad W. 2000. *Asymptotic Statistics*. Vol. 3. Cambridge university press.
Whitehead, John. 1980. "Fitting Cox's Regression Model to Survival Data Using GLIM." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 29 (3): 268–75.
Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.
Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Methodological)* 67 (2): 301–20.

*Sahir Rai Bhatnagar**
McGill University
2001 McGill College Avenue Montreal, QC, Canada H3A 1G1
http://sahirbhatnagar.com/
sahir.bhatnagar@mcgill.ca

*Maxime Turgeon**
University of Manitoba
186 Dysart Road Winnipeg, MB, Canada R3T 2N2
https://maxturgeon.ca/
max.turgeon@umanitoba.ca

Jesse Islam
McGill University
2001 McGill College Avenue Montreal, QC, Canada H3A 1G1
jesse.islam@mail.mcgill.ca

James A. Hanley
McGill University
2001 McGill College Avenue Montreal, QC, Canada H3A 1G1
http://www.medicine.mcgill.ca/epidemiology/hanley/
james.hanley@mcgill.ca

Olli Saarela
University of Toronto
Dalla Lana School of Public Health, 155 College Street, 6th floor, Toronto, Ontario M5T 3M7, Canada
http://individual.utoronto.ca/osaarela/
olli.saarela@utoronto.ca

dbcsp: User-friendly R package for Distance-Based Common Spatial Patterns

by Itsaso Rodríguez, Itziar Irigoien, Basilio Sierra, and Concepción Arenas

Abstract Common Spatial Patterns (CSP) is a widely used method to analyse electroencephalography (EEG) data, concerning the supervised classification of the activity of brain. More generally, it can be useful to distinguish between multivariate signals recorded during a time span for two different classes. CSP is based on the simultaneous diagonalization of the average covariance matrices of signals from both classes and it allows the data to be projected into a low-dimensional subspace. Once the data are represented in a low-dimensional subspace, a classification step must be carried out. The original CSP method is based on the Euclidean distance between signals, and here we extend it so that it can be applied on any appropriate distance for data at hand. Both the classical CSP and the new Distance-Based CSP (DB-CSP) are implemented in an R package, called **dbcsp**.

1 Background

Eigenvalue and generalized eigenvalue problems are very relevant techniques in data analysis. The well-known Principal Component Analysis with the eigenvalue problem in its roots was already established by the late seventies (Mardia et al., 1979). In mathematical terms, Common Spatial Patterns (CSP) is based on the generalized eigenvalue decomposition or the simultaneous diagonalization of two matrices to find projections in a low dimensional space. Although in algebraic terms PCA and CSP share several similarities, their main aims are different: PCA follows a non-supervised approach but CSP is a two-class supervised technique. Besides, PCA is suitable for standard quantitative data arranged in 'individuals × variables' tables, while CSP is designed to handle multivariate signals time series. That means that, while for PCA each individual or unit is represented by a classical numerical vector, for CSP each individual is represented by several signals recorded during a time span, i.e., by a 'number of signals × time span' matrix. CSP allows the individuals to be represented in a dimension reduced space, a crucial step given the high dimensional nature of the original data. CSP computes the average covariance matrices of signals from the two classes to yield features whose variances are optimal to discriminate the classes of measurements. Once data is projected into a low dimensional space, a classification step is carried out. The CSP technique was first proposed under the name Fukunaga-Koontz Transform in Fukunaga and Koontz (1970) as an extension of PCA, and Müller-Gerking et al. (1999) used it to discriminate electroencephalography data (EEG) in a movement task. Since then, it has been a widely used technique to analyze EEG data and develop Brain Computer Interfaces (BCI), with different variations and extensions (Blankertz et al., 2007a,b; Grosse-Wentrup and Buss, 2008; Lotte and Guan, 2011; Wang et al., 2012; Astigarraga et al., 2016; Darvish Ghanbar et al., 2021). In Wu et al. (2013), subject specific best time window and number of CSP features are fitted through a two-level cross validation scheme within the Linear Discriminant classifier. Samek et al. (2014) offer a divergence-based framework including several extensions of CSP. As a general term, CSP filter maximizes the variance of the filtered or projected EEG signals of one class of movements while minimizing it for the signals of the other class. Similarly, it can be used to detect epileptic activities Khalid et al. (2016) or other brain activities. BCI systems can also be of great help to people who suffer from some disorders of cerebral palsy, or who suffer from other diseases or disabilities that prevent the normal use of their motor skills. These systems can considerably improve the quality of life of these people, for which small advances and changes imply big improvements. BCI systems can also contribute to human vigilance detection, connected with occupations involving sustained attention tasks. Among others, CSP and variations of it have been applied to the vigilance estimation task (Yu et al., 2019).

The original CSP method is based on the Euclidean distance between signals. However, as far as we know, a generalization allowing the use of any appropriate distance was not introduced. The aim of the present work is to introduce a novel Distance-Based generalization of it (DB-CSP). This generalization is of great interest, since these techniques can also offer good solutions in other fields where multivariate time series data arise beyond pure electroencephalography data (Poppe, 2010; Rodríguez-Moreno et al., 2020).

Although CSP in its classical version is a very well-known technique in the field of BCI, it is not implemented in R. In addition, as DB-CSP is a new extension of it, it is worth building an R package that includes both CSP and DB-CSP techniques. The package offers functions in a user-friendly way for the less familiar users of R but it also offers complete information about its objects so that reproducible analysis can be carried out and more advanced and customised analysis can be performed taking

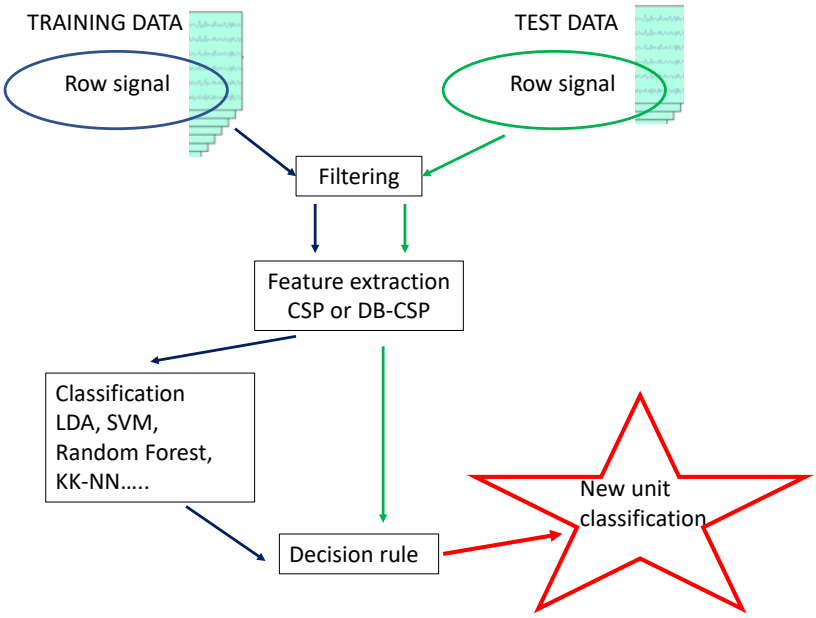


Figure 1: Flow-chart showing the steps to classify a new data. First, the filtering is done along with the feature extraction. This is the core of the procedure (CSP or DB-CSP). Then, a classifier is built to make the decision giving the classification of the new data.

advantage of already well-known packages of R.

The paper is organized as follows. First, we review the mathematical formulation of the Common Spatial Patterns method. Next, we present the core of our contribution describing both the novel CSP' extension based on distances and the **dbcsp** package. Then, the main functions in **dbcsp** are introduced along with reproducible examples of their use. Finally, some conclusions are drawn.

2 CSP and DB-CSP

Let us consider that we have n statistical individuals or units classified in two classes C_1 and C_2, with $\#C_1 = n_1$ and $\#C_2 = n_2$. For each unit i in class C_k, data from c sources or signals are collected during T time units and therefore unit i is represented in matrix the X_{ik} ($i = 1, \ldots, n_k$; $k = 1, 2$). For instance, for electroencephalograms, data are recorded by a c-sensor cap each t time units ($t = 1, \ldots, T$). As usual, we consider that each X_{ik} is already scaled or with the appropriate pre-processing in the context of application; for instance, if working with EGG data, each signal should be band-pass filtered before its use.

The goal is to classify a new unit X in C_1 or C_2. To this end, first a projection into a low-dimensional subspace is carried out. Then, as a standard approach the Linear Discriminant classifier (LDA) is applied taking as input data for the classifier the log-variance of the projections obtained in the first step. It is obvious that the importance of the technique lies mainly in the first step, and once it is done, LDA or any other classifiers could be applied. Based on that, we focus on how this projection into a low-dimensional space is done, from the classical CSP point of view as well as its novel extension DB-CSP (see Figure 1).

Classical CSP

The main idea is to use a linear transform to project or filter data into low-dimensional subspace with a projection matrix, in such a way that each row consists of weights for signals. This transformation maximizes the variance of two-class signal matrices. The method performs a simultaneous diagonalization of the covariance matrices of both classes. Given data $X_{11}, \ldots, X_{n_1 1}$ (matrices $c \times T$) from class C_1 and $X_{12}, \ldots, X_{n_2 2}$ (also matrices $c \times T$) from class C_2, the following steps are needed:

- All matrices are standardized so that traces of $X_{ik}X_{ik}'$ are the same.
- Compute average covariance matrices:

$$B_k = \frac{1}{n_k}\sum_{i=1}^{n_k} X_{ik}X_{ik}', \quad k = 1, 2$$

- Look for directions $W = (\mathbf{w}_1, \ldots, \mathbf{w}_c) \in \mathbb{R}^{c\times c}$ according to the criterion:

$$\begin{aligned} &\text{Maximize } tr(W'B_1W) \\ &\text{subject to } W'(B_1 + B_2)W = I \end{aligned}$$

The solution is given by the generalized spectral decomposition $B_1\mathbf{w} = \lambda B_2\mathbf{w}$ choosing the first and the last q eigenvectors: $W_{CSP} = (\mathbf{w}_1, \ldots, \mathbf{w}_q, \mathbf{w}_{c-q+1}, \ldots, \mathbf{w}_c)$.

Vectors $\mathbf{w}_j$ offer weights so that new signals $X_{i1}'\mathbf{w}_j$ and $X_{i2}'\mathbf{w}_j$ have big and low variability for the first q vectors ($j = 1, \ldots, q$) respectively, and vice-versa for the last q vectors ($j = c - q + 1, \ldots, c$). To clarify the notation and interpretation, let us denote $\mathbf{a}_j = \mathbf{w}_j$ the first q vectors and $\mathbf{b}_j = \mathbf{w}_{c+1-j}$ the last q. That way, and broadly speaking, variability of elements in C_1 is big when projecting on vectors $\mathbf{a}_j$ and low on vectors $\mathbf{b}_j$, and vice-versa, for elements in class C_2.

Finally, the log-variability of these new and few $2q$ signals are considered as input for the classification, which classically is the Linear Discriminant Analysis (LDA). Obviously, any other classification technique can be used, as it is illustrated in the subsection **Extending the example**.

Distance-based CSP

Following the commented ideas, the Distance-Based CSP (DB-CSP) is an extension of the classical CSP method. In the same way as the classical CSP, DB-CSP gives some weights to the original sources or signals and obtains new and few $2q$ signals which are useful for the discrimination between the two classes. Nevertheless, the considered distance between the signals can be any other than the Euclidean. The steps are the following:

- Compute an appropriate distance measure between sources and the double-centered inner product:

$$X_{ik} \to D_{ik} \to P_{ik} = -1/2HD_{ik}^{(2)}H, \quad i = 1, \ldots, n_k;\ k = 1, 2$$

where H stands for the centering matrix and the superindex in brackets (2) for squared elements in the matrix. Again, all matrices are standardized so that all traces of $X_{ik}X_{ik}'$ are the same.

- Compute average distance-based covariance matrices:

$$B_k^* = \frac{1}{n_k}\sum_{i=1}^{n_k}\left(P_{ik}P_{ik}' + X_{ik}\mathbf{x}_{ik}\mathbf{1}' + \mathbf{1}\mathbf{x}_{i,k}'X_{ik}' - \mathbf{x}_{ik}'\mathbf{x}_{ik}\mathbf{1}\mathbf{1}'\right)$$

where $\mathbf{x}_{ik} = \frac{1}{c}\mathbf{1}'\mathbf{X}_{ik}$, and $k = 1, 2$.

Once we have the covariance matrices related to the chosen distance matrix, the directions are found as in classical CSP and new signals $X_{ik}'\mathbf{a}_j$, $X_{ik}'\mathbf{b}_j$ are built ($j = 1, \ldots, q$). Again, for individuals in class C_1 the projections on vectors $\mathbf{a}$ and $\mathbf{b}$ are big and low respectively; for individuals in class C_2 it is the other way round.

It is important to note that if the chosen distance does not produce a positive definite covariance matrix, it must be replaced by a similar one that is positive definite.

When the selected distance is the Euclidean, then, **DB-CSP reduces to classical CSP**.

Once the q directions $\mathbf{a}_j$ and $\mathbf{b}_j$ are calculated, new $2q$ signals are built. Many interesting characteristics of the new signals could be extracted, although the most important in the procedure is the variance. Those characteristics of the new signals are the input data for the classification step.

3 Implementation

In this section, the structure of the package and the functions implemented are explained. The **dbcsp** package was developed for the free statistical R environment and it is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/dbcsp/index.html.

Input

The input data are the corresponding n_1 and n_2 matrices X_{ik} of the n units classified in classes C_1 and C_2, respectively ($i = 1, \ldots, n_k \; ; \quad k = 1, 2$). Let `x1` and `x2` be two lists of length n_1 and n_2, respectively, with X_{ik} matrices ($c \times T$) as elements of the lists. `NA` values are allowed. They are imputed by interpolating with the surrounding values via the `na.approx` function in package **zoo**. To ensure the user is aware of the missing values and their imputation, a warning is printed. We also consider that new items to be classified are in list `xt`. The aforementioned first step of the method is carried out by building the object called `"dbcsp"`.

`dbcsp` object

The `dbcsp` object is an S4 class created to compute the projection vectors W. The object has the following slots:

- **Slots**
  `X1 = "list"`, `X2 = "list"`, the lists `X1` and `X2` (lengths n_1 and n_2) containing the matrices X_{ik} for the two classes C_1 and C_2, respectively ($i = 1, \ldots, n_k \, ; \; k = 1, 2$).

  `q = "integer"`, to determine the number of pairs of eigenvectors $\mathbf{a}_j$ and $\mathbf{b}_j$ that are kept. By default `q=15`.

  `labels = "character"`, vector of two strings indicating labels names, by default names of elements in `X1` and `X2`.

  `type = "character"`, to set the type of distance to be considered, by default `type='EUCL'`. The supported distances are these ones:

  - Included in **TSdist**: `infnorm,ccor,sts,...`
  - Included in **parallelDist**: `bhattacharyya,bray,...`
  - Custom distances: it is also possible to use a user-defined distance function, a function `dcustom` which returns a scalar providing the distance value ($d(\mathbf{x}_{ik}, \mathbf{x}_{jk})$) between signals $\mathbf{x}_{ik}$ and $\mathbf{x}_{jk}$ ($i, j = 1, \ldots, n_k \, , \; k = 1, 2$). The name of the custom distance function is passed as character to the type parameter (`type="dcustom"`). The parallelDist package also allows the use of custom distances, but the distance function has to be defined using the `cppXPtr` function of the **RcppXPtrUtils** package, as is explained in the *User-defined distance functions* section of the parallelDist package documentation.

  `mixture = "logical"`, logical value indicating whether to use mixture of distances or not (EUCL + other), by default `mixture=FALSE`.

  `w = "numeric"`, weight for the mixture of distances $D_{\text{mixture}} = wD_{\text{euclidea}} + (1 - w)D_{\text{type}}$, by default `w=0.5`.

  `training = "logical"`, logical value indicating whether or not to perform the classification, by default `classification=FALSE`. If `classification=TRUE`, LDA discrimination based on the log-variances of the projected sources is considered, following the classical approach in CSP.

  `fold = "integer"`, integer value, by default `fold=10`. It controls the number of partitions for the k-fold validation procedure, if the classification is done.

  `seed = "numeric"`, numeric value, by default `seed=NULL`. Set a seed in case you want to be able to replicate the results.

`eig.tol = "numeric"`, numeric value, by default `eig.tol=1e-06`. If the minimum eigenvalue is below this tolerance, average covariance matrices are replaced by the most similar matrix that is positive definite. It is done via function `nearPD` in **Matrix** and a warning message is printed to make the user aware of it.

`out = "list"`, list containing elements of the output. Mainly, matrix W with vectors $\mathbf{a}_j$ and $\mathbf{b}_j$ in element `vectors`, log-variances of filtered signals in `proy` and partitions considered in the k-fold approach with reproducibility purposes.

- **Usage**
  Following the standard procedure in `R`, an instance of a class `dbcsp` is created via the `new()` constructor function:

```
new("dbcsp",X1 = x1,X1 = x2)
```

Slots `X1` and `X2` are compulsory since they contain the original data. When a slot is not specified, the default value is considered. First, the S4 object of class `dbcsp` must be created. By default, the Euclidean distance is used, nevertheless it can be changed. For instance, "Dynamic Transform Distance" (Giorgino et al., 2009) can be set:

```
mydbcsp <- new('dbcsp', X1=x1, X2=x2, type='dtw')
```

or a mixture between this distance and the Euclidean can be indicated by:

```
mydbcsp.mix <- new('dbcsp', X1=x1, X2=x2, labels=c("C1", "C2"),
                   mixture=TRUE, w=0.4,type="dtw")
```

Besides, a custom distance function can be defined and used when creating the object:

```
fn <- function(x, y) mean(1 - cos(x - y))
mydbcsp <- new("dbcsp", X1 = x, X2 = y, type="fn")
```

It is worth mentioning that it is possible to reduce the computational time through `parallelDist` custom distance option, where the function is defined using C++ and by creating an external pointer to the function by means of the `cppXPtr` function:

```
customEucli <- RcppXPtrUtils::cppXPtr(
                   "double customDist(const arma::mat &A, const arma::mat &B) {
                          return sqrt(arma::accu(arma::square(A - B)));
                   }",
                   depends = c("RcppArmadillo")
               )
mydbcsp <- new('dbcsp',x1,x2,type="customEucli")
```

The object contains all the information to carry out the classification task in a lower dimension space.

Functions `plot` and `boxplot`

For exploratory and descriptive purposes, the original signals X_{ik} and the projected ones can be plotted for the selected individual i in class k, and the selected pair of dimensions $\mathbf{a}_j$ and $\mathbf{b}_j$ ($i = 1, \ldots, n_k$, $k = 1, 2$).

- **Usage**
  `plot(mydbcsp)`

- **Arguments**
  `x`, an object of class `dbcsp`

  `class`, integer to indicate which of both classes to access (1 or 2), by default `class=1`.

  `index`, integer to indicate which instance of the class to plot, by default `index=1`.

`vectors`, integer to indicate which j projected signals are to be plotted. By default all the vectors used in the projection are plotted.

`pairs` logical, if `TRUE` the pairs $\mathbf{a}_j$ and $\mathbf{b}_j$ of the indicated indices are also shown, by default `pairs=TRUE`.

`before` logical, if `TRUE` the original signals are plotted, by default `before=TRUE`.

`after` logical, if `TRUE` the signals after projection are plotted, by default `after=TRUE`.

`legend` logical, if `TRUE`, a legend for filtered signals is shown, by default `legend=FALSE`.

`getsignals` logical, if `TRUE`, the projected signals are returned.

Besides, the log-variances of the projected signals of both classes can be shown in boxplots. This graphic can help to understand the discriminative power that is in the low-dimension space.

- **Usage**
  `boxplot(mydbcsp)`

- **Arguments**
  `x`, an object of class `dbcsp`

  `vectors`, integer or vector of integers, indicating the index of the projected vectors to plot, by default `index=1`.

  `pairs` logical, if `TRUE` the pairs $\mathbf{a}_j$ and $\mathbf{b}_j$ of the indicated indices are also shown, by default `pairs=TRUE`.

  `show_log` logical, if `TRUE` the logarithms of the variances are displayed, otherwise the variances, by default `show_log=TRUE`.

It is worth taking into account that in the aforementioned functions, values in argument `vectors` must lie between 1 and $2q$, being q the number of dimensions used to perform the DB-CSP algorithm when creating the `dbcsp` object. Therefore, values 1 to q correspond to vectors $\mathbf{a}_1$ to $\mathbf{a}_q$ and values $q+1$ to $2q$ correspond to vectors $\mathbf{b}_1$ to $\mathbf{b}_q$. Then, if `pairs=TRUE`, it is recommended that values in argument `vectors` are in $\{1,\ldots,q\}$, since their pairs are plotted as well. When values are above q, it should be noted that they correspond to vectors $\mathbf{b}_1$ to $\mathbf{b}_q$. For instance, if `q=15` and `boxplot(object,vectors=16,pairs=FALSE)`, vector $\mathbf{b}_1$ $(16-q=1)$ is shown.

Function `selectQ`, Function `train` and Function `predict`

The functions in this section help the classification step in the procedure. Function `selectQ` helps to find an appropriate dimension needed for the classification. Given different values of dimensions, the accuracy related to each dimension is calculated so that the user can assess which dimension of the reduced space can be sufficient. A k-fold cross-validation approach or a holdout approach can be followed. Function `train` performs the Linear Discriminant classification based on the log-variances of the dimensions built in the `dbcsp` object. Since LDA has a geometric interpretation that makes the classifier sensible for more general situations Duda et al. (2001), not the normality nor the homoscedasticity of data are checked. The accuracy of the classifier is computed based on the k-fold validation procedure. Finally, function `predict` performs the classification of new individuals.

- **Usage of `selectQ`**
  `selectQ(mydbcsp)`

- **Arguments**
  `object`, an object of class `dbcsp`

`Q`, vector of integers which represents the dimensions to use, by default `Q=c(1,2,3,5,10,15)`.

`train_size`, float between 0.0 and 1.0 representing the proportion of the data set to include in the train split, by default `train_size=0.75`.

`CV`, logical indicating whether a *k*-fold cross validation must be performed or a hold-out approach (if `TRUE`, `train_size` is not used), by default `CV=FALSE`.

`folds` integer, number of folds to use if `CV` is performed.

`seed` numeric value, by default `seed=NULL`. Set a seed in case you want to be able to replicate the results.

This function returns the accuracy values related to each dimension set in `Q`. If `CV=TRUE`, the mean accuracy as well as the standard deviation among folds is also returned.

- **Usage of** `train`
  `train(mydbcsp)` or embedded as a parameter in:
  `new('dbcsp',X1=x1,X2=x2,training=TRUE,type="dtw")`

- **Arguments**
  `x`, an object of class `dbcsp`

  `selected_q`, integer value indicating the number of vectors to use when training the model. By default all dimensions considered when creating the object `dbcsp`.

  Besides, arguments `seed` and `fold` are available.

It is important to note that in this way a classical analysis can be carried out, in the sense of:

- LDA is applied based on the log-variances of the dimensions indicated by the user in `select_q`;
- percentage of correct classification is obtained via *k*-fold cross validation.

However, it is evident that it may be of interest to use other classifiers or other characteristics in addition to or different from log-variances. This more advanced procedure is explained below. See the basic analysis of the **User guide with a real example** section in order to visualize and follow the process of a first basic/classic analysis.

- **Usage of** `predict`
  `predict(mydbcsp,X_test=xt)`

- **Arguments**
  `object`, an object of class `dbcsp`

  `X_test`, list of matrices to be classified.

  `true_targets`, optional, if available, vector of true labels of the instances. Note that they must match the name of the labels used when training the model.

4 User guide with a real example

To show an example beyond pure electroencephalography data, Action Recognition data is considered. Besides having a reproducible example to show the use of the implemented functions and the results they offer, this Action Recognition data set is included in the package. The data set contains the skeleton data extracted from videos of people performing six different actions, recorded by a semi-humanoid robot. It consists of a total of 272 videos with 6 action categories. There are around 45 clips in each category, performed by 46 different people. Each instance is composed of 50 signals (xy coordinates for 25 body key points extracted using OpenPose (Cao et al., 2019)), where each signal has 92 values, one per frame. These are the six categories included in the data set:

1. Come: gesture for telling the robot to come to you. There are 46 instances for this class.
2. Five: gesture of 'high five'. There are 45 instances for this class.
3. Handshake: gesture of handshaking with the robot. There are 45 instances for this class.
4. Hello: gesture for saying hello to the robot. There are 44 instances for this class.
5. Ignore: ignore the robot, pass by. There are 46 instances for this class.
6. Look at: stare at the robot in front of it. There are 46 instances for this class.

The data set is accessible via `AR.data` and more specific information can be found in (Rodríguez-Moreno et al., 2020). Each class is a list of matrices of $[K \times num\_frames]$ dimensions, where $K = 50$ signals and $num\_frames = 92$ values. As mentioned before, the 50 signals represent the xy coordinates of 25 body key points extracted by OpenPose.

For example, two different classes can be accessed this way:

```
x1 <- AR.data$come
x2 <- AR.data$five
```

where, `x1` is a list of 46 instances of $[50x92]$ matrices of *come* class and `x2` is a list of 45 instances of $[50x92]$ matrices of *five* class. An example of skeleton sequences for both classes is shown in Figure 2 (left, for class *come* and right, for class *five*).

Figure 2: Sequences of the skeleton extracted from the videos. Left: sequence for action 'come'. Right: sequence for action '(high) five'. For each frame, x and y coordinates of the 25 body key points of the skeleton are extracted by OpenPose.

Next, the use of functions in `dbcsp` is shown based on this data set. First a basic/classic analysis is performed.

Basic/classic analysis

Let us consider an analysis using 15-dimensional projections and the Euclidean distance. At a first step the user can obtain vectors W by:

```
x1 <- AR.data$come
x2 <- AR.data$five
mydbcsp <- new('dbcsp', X1=x1, X2=x2, q=15, labels=c("C1", "C2"))
summary(mydbcsp)
```

Creating the object `mydbcsp`, the vectors W are calculated. As indicated in parameter `q=15`, the first and last 15 eigenvectors are retained. With `summary`, the obtained output is:

```
There are 46 instances of class C1 with [50x92] dimension.
There are 45 instances of class C2 with [50x92] dimension.
The DB-CSP method has used 15 vectors for the projection.
EUCL distance has been used.
Training has not been performed yet.
```

Now, if the user knows from the beginning that 3 is an appropriate dimension, the classification step could be done while creating the object. Using classical analysis, with for instance 10-fold, LDA as classifier and log-variances as characteristics, the corresponding input and summary output are:

```
mydbcsp <- new('dbcsp', X1=x1, X2=x2, q=3, labels=c("C1", "C2"), training=TRUE, fold = 10, seed = 19)
summary(mydbcsp)
```

```
There are 46 instances of class C1 with [50x92] dimension.
There are 45 instances of class C2 with [50x92] dimension.
The DB-CSP method has used 3 vectors for the projection.
EUCL distance has been used.
An accuracy of 0.9130556 has been obtained with 10 fold cross validation and using 3 vectors when training.
```

If a closer view of the accuracies among the folds is needed, the user can obtain them from the `out` slot of the object:

```
# Accuracy in each fold
mydbcsp@out$folds_acc

# Intances belonging to each fold
mydbcsp@out$used_folds
```

Basic/classic analysis selecting the value of q

Furthermore, it is clear that the optimal value of q should be chosen based on the percentages of correct classification. It is worth mentioning that the LDA is applied on the $2q$ projections, as set in the object building step. It is interesting to measure how many dimensions would be enough using `selectQ` function:

```
mydbcsp <- new('dbcsp', X1=x1, X2=x2, labels=c("C1", "C2"))
selectDim <- selectQ(mydbcsp, seed=30, CV=TRUE, fold = 10)

selectDim
   Q       acc         sd
1  1 0.7663889 0.12607868
2  2 0.9033333 0.09428818
3  3 0.8686111 0.11314534
4  5 0.8750000 0.13289537
5 10 0.8797222 0.09513230
6 15 0.8250000 0.05257433
```

Since the 10-fold cross-validation approach is chosen, the mean accuracies as well as the corresponding standard deviations are returned. Thus, with Linear Discriminant Analysis (LDA), log-variances as characteristics, it seems that dimensions related to first and last $q = 2$ eigenvectors (2×2 dimensions in total) are enough to obtain a good classification, with an accuracy of 90%. Nevertheless, it can also be observed that variation among folds can be relevant.

To visualize what is the representation in the reduced dimension space function `plot` can be used. For instance, to visualize the first unit of the first class, based on projections along the 2 first and last vectors ($\mathbf{a}_1, \mathbf{a}_2$ and $\mathbf{b}_1, \mathbf{b}_2$):

```
plot(mydbcsp, index=1, class=1, vectors=1:2)
```

In the top graphic of Figure 3, the representation of the first video of class C_1 given by non standardized matrix X_{11} can be seen, where the horizontal axis represents the frames of the video and the lines are the positions of the body key points (50 lines). In the bottom graphic, the same video is represented in a reduced space where the video is represented by the new signals (only 4 lines).

To have a better insight of the discriminating power of the new signals in the reduced dimension space, we can plot the corresponding log-variances of the new signals. Parameter `vectors` in function `boxplot` sets which are the eigenvectors considered to plot.

```
boxplot(mydbcsp, vectors=1:2)
```

In Figure 4 it can be seen that variability of projections on the first eigenvector direction ($\log(VAR(X'_{ik}\mathbf{a}_1))$) are big for elements in `x1`, but small for elements in `x2`. Analogously, projecting on the last dimension ($\log(VAR(X'_{ik}\mathbf{b}_1))$), low variability is held in `x1` and big variability in `x2`. The same pattern holds when projecting on vectors $\mathbf{a}_2$ and $\mathbf{b}_2$.

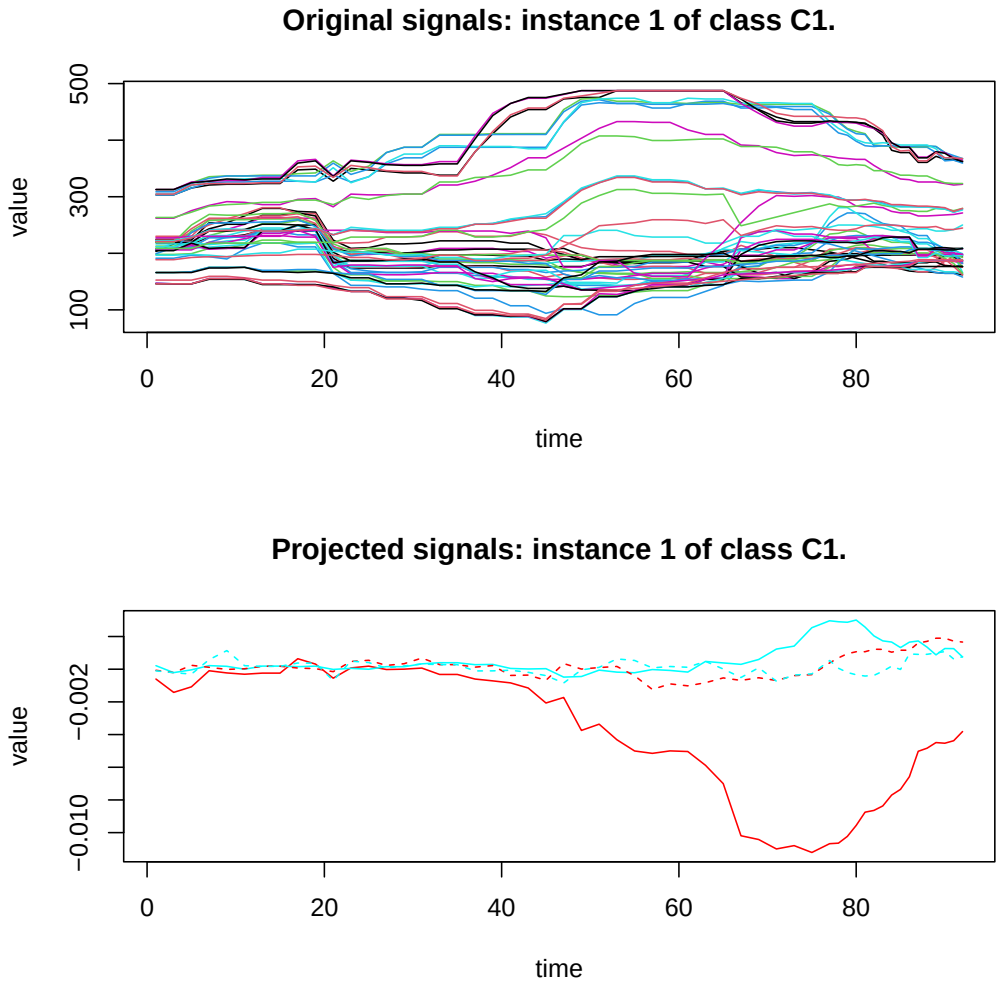


Figure 3: Representation of the first video of class C_1. Top: original version where each line corresponds to the signal of a body key point. Bottom: the projections on vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ (continuous lines) and $\mathbf{b}_1$ and $\mathbf{b}_2$ (dotted lines). Being a video of class C_1, variabilities of the projections on vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ are big whereas on vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are small, as expected.

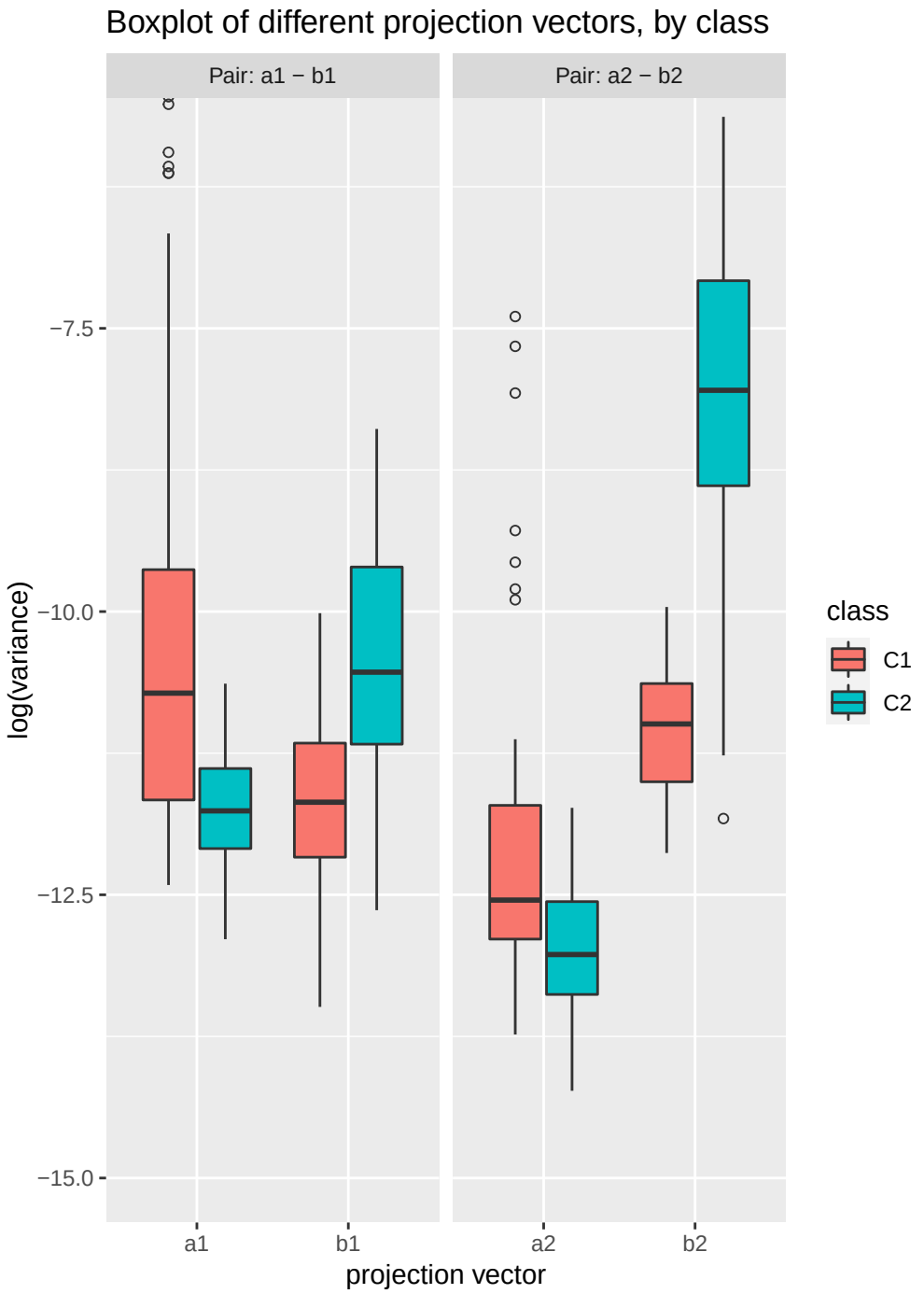


Figure 4: Log-variabilities of the projected signals on vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ and $\mathbf{b}_1$ and $\mathbf{b}_2$, separated by classes C_1 and C_2. By construction, variabilities of the projections on vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ are big for units in class C_1 and small for units C_2; opposite pattern can be seen for projections on vectors $\mathbf{b}_1$ and $\mathbf{b}_2$.

Basic/classic analysis new unit classification

Once the value of q has been decided and the accuracy of the classification is known, the classifier should be built (through `train()`) so that the user can proceed to predict the class a new action held in a video belongs to, using the function `predict`. For instance, with only illustrative purpose, we can classify the first 5 videos which are stored in `x1`.

```
mydbcsp <- train(mydbcsp, selected_q=2, verbose=FALSE)
xtest <- x1[1:5]
outpred <- predict(mydbcsp, X_test=xtest)
```

If the labels of the testing items are known, the latter function returns the accuracy.

```
outpred <- predict(mydbcsp, X_test=xtest, true_targets= rep("C1", 5))
```

Finally, notice that the user could use any other distance instead of the Euclidean between the signals to compute the important directions $\mathbf{a}_j$ and $\mathbf{b}_j$. For instance, in this case it could be appropriate to use the Dynamic Time Warping distance, setting so in the argument `type="dtw"`:

```
# Distance DTW
mydbcsp.dtw <- new('dbcsp', X1=x1, X2=x2, labels=c("C1", "C2"), type="dtw")
```

5 Extending the example

In the previous section a basic workflow to use functions implemented in **dbcsp** is presented. Nevertheless, it is straightforward to extend the procedure. Once the interesting directions in W are calculated through `dbcsp`, other summarizing characteristics beyond the variance could be extracted from the projected signals, as well as other classifiers which could be used in the classification step. For those purposes, `dbcsp` is used to compute the directions in W that will be the base to calculate other features as well as the input features for other classifiers. Here it is shown how, once the eigenvectors are extracted from an object `dbcsp`, several characteristics could be extracted from the signals and a new `data.frame` can be built so that any other classification technique could be applied. In this example we worked with **caret** package to apply different classifiers. It is important to pay attention to which the train and test sets are, so that the vectors are computed based only on training set instances.

```
# Establish training and test data
n1 <- length(x1)
trainind1 <- rep(TRUE, n1)
n2 <- length(x2)
trainind2 <- rep(TRUE, n2)
set.seed(19)
trainind1[sample(1:n1, 10, replace=FALSE)] <- FALSE
trainind2[sample(1:n2, 10, replace=FALSE)] <- FALSE
x1train <- x1[trainind1]
x2train <- x2[trainind2]

# Extract the interesting directions
vectors <- new('dbcsp', X1=x1train, X2=x2train, q=5, labels=c("C1", "C2"))@out$vectors

# Function to calculate the desired characteristics from signals
calc_info <- function(proj_X, type){
  values <- switch(type,
                   'var' = values <-  plyr::laply(proj_X, function(x){apply(x,1,var)}),
                   'max' = values <-  plyr::laply(proj_X, function(x){apply(x,1,max)}),
                   'min' = values <- plyr::laply(proj_X, function(x){apply(x,1,min)}),
                   'iqr' = values <- plyr::laply(proj_X, function(x){
                     apply(x,1,function(y){
                       q <- quantile(y, probs = c(0.25, 0.75))
                       q[2] -q[1]
                     })
                    })
  )
  return(values)
}
```

By means of this latter function, besides the variance of the new signals, the maximum, the minimum, and the interquartile range can be extracted.

Next, imagine we want to perform our classification step with the interquartile range information along with the log-variance.

```
# Project units of class C1 and
projected_x1 <- plyr::llply(x1, function(x,W) t(W)%*%x, W=vectors)

# Extract the characteristics
logvar_x1 <- log(calc_info(projected_x1,'var'))
iqr_x1 <- calc_info(projected_x1,'iqr')
new_x1 <- data.frame(logvar=logvar_x1, iqr=iqr_x1)

# Similarly for units of class C2
projected_x2 <- plyr::llply(x2, function(x,W) t(W)%*%x, W=vectors)
logvar_x2 <- log(calc_info(projected_x2,'var'))
iqr_x2 <- calc_info(projected_x2,'iqr')
new_x2 <- data.frame(logvar=logvar_x2, iqr=iqr_x2)


# Create dataset for classification
labels <- rep(c('C1','C2'), times=c(n1,n2))
new_data <- rbind(new_x1,new_x2)
new_data$label <- factor(labels)
new_data_train <- new_data[c(trainind1, trainind2), ]
new_data_test <- new_data[!c(trainind1, trainind2), ]

# Random forest
trControl <- caret::trainControl(method = "none")
rf_default <- caret::train(label~.,
                           data = new_data_train,
                           method = "rf",
                           metric = "Accuracy",
                           trControl = trControl)
rf_default

# K-NN
knn_default <- caret::train(label~.,
                            data = new_data_train,
                            method = "knn",
                            metric = "Accuracy",
                            trControl = trControl)
knn_default

# Predictions and accuracies on test data
# Based on random forest classifier
pred_labels <- predict(rf_default, new_data_test)
predictions_rf <- caret::confusionMatrix(table(pred_labels,new_data_test$label))
predictions_rf

# Based on knn classifier
pred_labels <- predict(knn_default, new_data_test)
predictions_knn <- caret::confusionMatrix(table(pred_labels,new_data_test$label))
predictions_knn
```

Thus, it is easy to integrate results and objects that **dbcsp** builds so that they can be integrated with other R packages and functions. This is interesting for more advanced users to perform their own customized analysis.

6 Conclusions

In this work a new Distance-Based Common Spatial Pattern is introduced. It allows to perform the classical Common Spatial Pattern when the Euclidean distance between signals is considered, but

it can be extended to the use of any other appropriate distance between signals as well. All of it is included in package the **dbcsp**. The package is easy to use for non-specialised users but, for the sake of flexibility, more advanced analysis can be carried out combining the created object and obtained results with already well-known R packages, such as **caret**, for instance.

Acknowledgements

This research was partially supported: IR by The Spanish Ministry of Science, Innovation and Universities (FPU18/04737 predoctoral grant). II by the Spanish Ministerio de Economia y Competitividad (RTI2018-093337-B-I00; PID2019-106942RB-C31). CA by the Spanish Ministerio de Economia y Competitividad (RTI2018-093337-B-I00, RTI2018-100968-B-I00) and by Grant 2017SGR622 (GRBIO) from the Departament d'Economia i Coneixement de la Generalitat de Catalunya. BS II by the Spanish Ministerio de Economia y Competitividad (RTI2018-093337-B-I00).

Author's contributions

II and CA designed the study. IR and II wrote and debugged the software. IR, II and CA checked the software. II, CA, IR and BS wrote and reviewed the manuscript. All authors have read and approved the final manuscript.

Bibliography

A. Astigarraga, A. Arruti, J. Muguerza, R. Santana, J. I. Martin, and B. Sierra. User adapted motor-imaginary brain-computer interface by means of EEG channel selection based on estimation of distributed algorithms. *Mathematical Problems in Engineering*, page 1435321, 2016. URL https://doi.org/10.1155/2016/1435321. [p80]

B. Blankertz, M. Kawanabe, R. Tomioka, F. U. Hohlefeld, V. V. Nikulin, and K.-R. Müller. Invariant common spatial patterns: Alleviating nonstationarities in brain-computer interfacing. In *NIPS'07: Proceedings of the 20th International Conference on Neural Information Processing*, pages 113–120, 2007a. [p80]

B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Muller. Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1):41–56, 2007b. URL https://doi.org/10.1109/MSP.2008.4408441. [p80]

Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. OpenPose: realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 (1):172–186, 2019. URL https://doi.org/10.1109/TPAMI.2019.2929257. [p86]

K. Darvish Ghanbar, T. Yousefi Rezaii, A. Farzamnia, and I. Saad. Correlation-based common spatial pattern (CCSP): A novel extension of CSP for classification of motor imagery signal. *PLOS ONE*, 16:1–18, 2021. doi: 10.1371/journal.pone.0248511. URL https://doi.org/10.1371/journal.pone.0248511. [p80]

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2001. [p85]

K. Fukunaga and W. L. Koontz. Application of the Karhunen-Loève expansion to feature selection and ordering. *IEEE Transactions on Computers*, 100(4):311–318, 1970. [p80]

T. Giorgino et al. Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of statistical Software*, 31(7):1–24, 2009. URL http://dx.doi.org/10.18637/jss.v031.i07. [p84]

M. Grosse-Wentrup and M. Buss. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE Transactions on Biomedical Engineering*, 55(8):1991–2000, 2008. URL https://doi.org/10.1109/TBME.2008.921154. [p80]

M. I. Khalid, T. Alotaiby, S. A. Aldosari, S. A. Alshebeili, M. H. Al-Hameed, F. S. Y. Almohammed, and T. S. Alotaibi. Epileptic MEG spikes detection using common spatial patterns and linear discriminant analysis. *IEEE Access*, 4:4629–4634, 2016. URL https://doi.org/10.1109/access.2016.2602354. [p80]

F. Lotte and C. Guan. Regularizing common spatial patterns to improve BCI designs: Unified theory and new algorithms. *Transactions on Biomedical Engineering*, 58(2):355–362, 2011. URL https://doi.org/10.1109/TBME.2010.2082539. [p80]

K. Mardia, J. Kent, and J. Bibby. *Multivariate Analysis*. Academic Press, London, 1979. [p80]

J. Müller-Gerking, G. Pfurtscheller, and H. Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Clinical Neurophysiology*, 110(5):787–798, 1999. URL https://doi.org/10.1016/S1388-2457(98)00038-8. [p80]

R. Poppe. Common spatial patterns for real-time classification of human actions. In *Machine Learning for Human Motion Analysis: Theory and Practice*, pages 55–73. IGI Global, 2010. [p80]

I. Rodríguez-Moreno, J. M. Martínez-Otzeta, I. Goienetxea, I. Rodriguez-Rodriguez, and B. Sierra. Shedding light on people action recognition in social robotics by means of common spatial patterns. *Sensors*, 20(8):2436, 2020. [p87]

I. Rodríguez-Moreno, J. M. Martínez-Otzeta, B. Sierra, I. Irigoien, I. Rodriguez-Rodriguez, and I. Goienetxea. Using common spatial patterns to select relevant pixels for video activity recognition. *Applied Sciences*, 10(22), 2020. URL https://www.mdpi.com/2076-3417/10/22/8075. [p80]

W. Samek, M. Kawanabe, and K.-R. Müller. Divergence-based framework for common spatial patterns algorithms. *IEEE Reviews in Biomedical Engineering*, 7:50–72, 2014. URL https://doi.org/10.1109/RBME.2013.2290621. [p80]

H. Wang, Q. Tang, and W. Zheng. L1-norm-based common spatial patterns. *IEEE Transactions on Biomedical Engineering*, 59(3):653–662, 2012. URL https://doi.org/10.1109/TBME.2011.2177523. [p80]

S.-L. Wu, C.-W. Wu, N. R. Pal, C.-Y. Chen, S.-A. Chen, and C.-T. Lin. Common spatial pattern and linear discriminant analysis for motor imagery classification. In *2013 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)*, pages 146–151. IEEE, 2013. [p80]

H. Yu, H. Lu, S. Wang, K. Xia, Y. Jiang, and P. Qian. A general common spatial patterns for EEG analysis with applications to vigilance detection. *IEEE Access*, 7:111102–111114, 2019. URL https://doi.org/10.1109/ACCESS.2019.2934519. [p80]

Itsaso Rodríguez
Department of Computation Science and Artificial Intelligence, University of the Basque Country UPV/EHU
Manuel Lardizabal 1, Donostia
Spain
itsaso.rodriguez@ehu.eus

Itziar Irigoien
Department of Computation Science and Artificial Intelligence, University of the Basque Country UPV/EHU
Manuel Lardizabal 1, Donostia
Spain
itziar.irigoien@ehu.eus

Basilio Sierra
Department of Computation Science and Artificial Intelligence, University of the Basque Country UPV/EHU
Manuel Lardizabal 1, Donostia
Spain
b.sierra@ehu.eus

Concepción Arenas
Department of Genetics, Microbiology and Statistics. Statistics Section, University of Barcelona UB
Diagonal 645, Barcelona
Spain
carenas@ub.edu

The R Package HDSpatialScan for the Detection of Clusters of Multivariate and Functional Data using Spatial Scan Statistics

by Camille Frévent, Mohamed-Salem Ahmed, Julien Soula, Zaineb Smida, Lionel Cucala, Sophie Dabo-Niang and Michaël Genin

Abstract This paper introduces the R package **HDSpatialScan**. This package allows users to easily apply spatial scan statistics to real-valued multivariate data or both univariate and multivariate functional data. It also permits plotting the detected clusters and to summarize them. In this article the methods are presented and the use of the package is illustrated through examples on environmental data provided in the package.

1 Introduction

Spatial cluster detection methods are useful tools for objective detection and localization of statistically significant aggregation of events indexed in space. Examples of the applications of these methods are numerous: in the field of epidemiology, these methods allow epidemiologists to detect spatial clusters of disease cases and to formulate etiological hypotheses; in the environmental sciences, researchers can be led to search for particularly polluted geographical areas, either by one pollutant in particular or by several pollutants simultaneously. In astronomy, researchers may want to identify star clusters from telescope image data.

Several cluster detection methods have been proposed in the literature. In particular, spatial scan statistics (originally proposed by Kulldorff and Nagarwalla (1995) and Kulldorff (1997) for Bernoulli and Poisson models) are powerful methods for detecting statistically significant spatial clusters, which can be defined by an aggregation of sites presenting an abnormal concentration (mean, etc) of an observed variable, with a variable scanning window and in the absence of pre-selection bias (objective detection of the cluster). Following on from Kulldorff's initial work, several researchers have adapted spatial scan statistics to other spatial data distributions: exponential (Huang et al., 2007), ordinal (Jung et al., 2007), normal (Kulldorff et al., 2009), Weibull (Bhatt and Tiwari, 2014), etc. Others use nonparametric approaches such as Jung and Cho (2015) and Cucala (2016) who respectively extend the Wilcoxon-Mann-Whitney test for spatial scan statistics and for temporal or spatial scan statistics. Note that in the case of spatial data the two approaches are equivalent by generalizing the method of Jung and Cho (2015) to detect either high or low clusters.

The applications of scan statistics are numerous. In the field of epidemiology, Khan et al. (2021) detected significant clusters of diabetes incidence in Florida between 2007 and 2010, which will help guide local health policies. Marciano et al. (2018) sought to detect spatial clusters of leprosy incidence in a hyperendemic Brazilian municipality between 2000 and 2005 and 2006 and 2010. The study showed a high percentage of contact between people which facilitates the transmission of the disease. Genin et al. (2020) detected high-risk clusters of Crohn's disease in France over the period 2007-2014. As the causes of this disease are still poorly understood, the detection of spatial clusters of Crohn's disease allows the researchers to make hypotheses on possible risk factors, such as high-social deprivation or high urbanization. In the context of environmental science, the detection of clusters of symptomatic exposure to pesticides in rural areas (Sudakin et al., 2002) would allow the monitoring and prevention of pesticide-related diseases. Gao et al. (2014) focused on the presence of iodine in drinking water in Shandong Province, China. The detection of spatial clusters of iodine presence in drinking water allows an improvement of the monitoring of drinking water quality in these geographical areas. Finally in the context of pollution data, Wan et al. (2020) and Shi et al. (2021) respectively detected clusters of high concentrations of $PM_{2.5}$ in America and China. Such results may allow local authorities to specifically monitor these areas and make decisions to reduce pollution.

When multiple variables are observed simultaneously at each spatial location, researchers may be interested in detecting spatial clusters with anomalous values of all measured variables. In this context, Kulldorff et al. (2007) proposed a multivariate spatial scan statistic using a combination of independent

univariate scan statistics. However it fails to take into account the potential correlations between the variables. A first spatial scan statistic for multivariate data taking into account the correlations was proposed by Cucala et al. (2017). Their method is based on a multivariate normal probability model and a likelihood ratio. Later, Cucala et al. (2019) proposed a nonparametric spatial scan statistic for multivariate data based on a multivariate Wilcoxon-Mann-Whitney test.

Technological developments in measurement tools and data storage capacity have yielded to the increasing use of sensors, cell phones and more generally connected devices that collect data continuously or almost continuously over time. This has led to the introduction of new analysis methods for functional data (Ramsay and Silverman, 2005), as well as the adaptation of classical statistical methods such as principal component analysis (Boente and Fraiman, 2000; Berrendero et al., 2011) or regression (Cuevas et al., 2002; Ferraty and Vieu, 2002; Chiou and Müller, 2007).
In the field of spatial scan statistics, Frévent et al. (2021a) and Smida et al. (2022) proposed new methods for univariate processes. However for example, in environmental surveillance, numerous variables are simultaneously measured, making a multivariate functional approach necessary to detect environmental black-spots. These can be defined as geographical areas characterized by elevated concentrations of multiple pollutants. Although Smida et al. (2022) only studied their approach in the univariate functional framework, they suggest that it could also be adapted for multivariate processes. Frévent et al. (2021b) studied this adaptation and also developed new efficient methods for multivariate functional spatial scan statistics.

In R several packages provide spatial scan statistics implementations. The best known is certainly the **rsatscan** (Kleinman, 2015) package which provides functions to interface R and the SaTScan software (Kulldorff, 2021), allowing the latter to be launched from R. It implements lots of univariate methods (ordinal, Bernoulli, Poisson, ...) but also the space-time spatial scan statistic (Kulldorff et al., 1998) and the multivariate extensions proposed by Kulldorff et al. (2007). The function `kulldorff` implemented in the R package **SpatialEpi** (Chen et al., 2018) also performs the spatial scan statistics based on the Poisson and the Bernoulli models. Other softwares were created to detect clusters such as ClusterSeer (Greiling et al., 2012; Durbeck et al., 2012) which performs spatial, temporal and space-time clustering, and TreeScan (Kulldorff, 2018) which implements the tree-based scan statistic (Kulldorff et al., 2003). We should also mention the R package **DCluster** (Gómez-Rubio et al., 2015) which implements the spatial scan statistics for Poisson or Bernoulli models. The R package **DClusterm** (Gómez-Rubio et al., 2019; Gomez-Rubio et al., 2020) also implements a cluster detection method. Briefly, it consists in considering a large number of generalized linear models by including potential cluster indicators one by one, and then to use a model selection procedure. The Shiny application SpatialEpiApp (Moraga, 2017b) and the R package **SpatialEpiApp** (Moraga, 2017a) allow the detection and visualization of clusters by using the scan statistics implemented in SaTScan. Finally the software FlexScan (Takahashi et al., 2010) and the R package **rflexscan** (Otani and Takahashi, 2021) implement the spatial scan statistic using a scan window with a non pre-defined shape, defined by Takahashi and Tango (2005). Other R packages also allow clusters detection such as **graphscan** (Loche et al., 2016) (the `cluster` function), **SPATCLUS** (Dematteï et al., 2006) or **scanstatistics** (Allévius, 2018b,a) for spatial or space-time data. It should be noted that these last two packages are no longer available on the CRAN (The Comprehensive R Archive Network) repository. Although existing packages implement a large number of statistical spatial scan models, none of them propose multivariate scan models taking into account the potential correlations between variables or scan models for functional data. Thus, we have developed the R package **HDSpatialScan** for high-dimensional spatial scan statistics. The latter allows on the one hand the detection of spatial clusters in multivariate or functional data, and on the other hand, their display on a map and the description of their characteristics.

This paper is organized as follows: The following section presents the different models implemented in the R package **HDSpatialScan**. Then, the implementation of the methods is described and examples of use of the package are given. The last section concludes the paper.

2 Models

Let $s_1, \ldots, s_n$ be n different locations of an observation domain $S \subset \mathbb{R}^2$ and $X_1, \ldots, X_n$ be the observations of a variable X in $s_1, \ldots, s_n$. Hereafter all observations are considered to be independent, which is a classical assumption in scan statistics. Three types of spatial data can be considered: either lattice data (the data are aggregated at the spatial level, e.g.: county), geostatistical data (the variable is defined on a continuous area and each individual measure corresponds to a unique fixed spatial location, e.g.: pollutant concentration measured by sensors over a region), or marked point data (each individual measure corresponds to a unique random spatial location, e.g.: height of the trees in a

forest, the location of the trees is random).

Spatial scan statistics aim at detecting spatial clusters and testing their statistical significance. Hence, one tests a null hypothesis $\mathcal{H}_0$ (the absence of a cluster) against a composite alternative hypothesis $\mathcal{H}_1$ (the presence of at least one cluster $w \subset S$ presenting abnormal values of X). For this purpose, a spatial scan statistic consists of two steps. The first one is a detection phase using a scanning window of variable size and shape. We will focus here on the approach of Kulldorff and Nagarwalla (1995) which use a circular scanning window of variable center and radius, however it should be noted that other shapes can be considered (Kulldorff et al., 2006; Cucala et al., 2013). An approach often advised is to limit the maximum size to half of the studied region since otherwise it would be like detecting a "negative cluster" in the areas outside the clusters covering almost all the studied region (Kulldorff and Nagarwalla, 1995). Then the scanning window allows to define a set of potential clusters $\mathcal{W}$ by

$$\mathcal{W} = \{w_{i,j} \ / \ 1 \leq |w_{i,j}| \leq \frac{n}{2}, \ 1 \leq i, j \leq n\}, \tag{1}$$

where $w_{i,j}$ is the disc centered on s_i that passes through s_j and $|w_{i,j}|$ corresponds to the number of sites in $w_{i,j}$. Figure 1 illustrates the set of potential clusters defined with a circular scanning window with Equation 1 on a set of eight administrative areas in France.

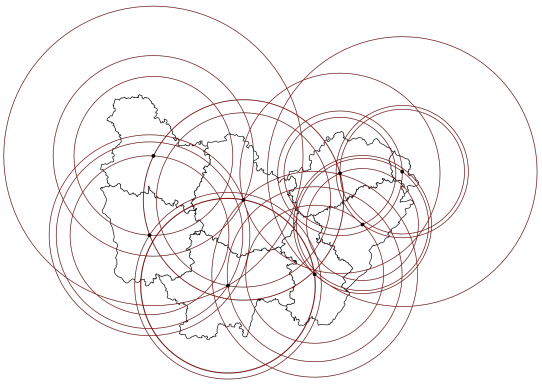


Figure 1: Set of potential clusters defined with a circular scanning window of variable center and radius with Equation 1 on a set of eight administrative areas in France. Each potential cluster is represented with a red circle.

Then the spatial scan statistic can be defined as the maximum of a concentration index over the set of potential clusters $\mathcal{W}$. The second step is the determination of the statistical significance of the spatial scan statistic. For this, since the distribution of the scan statistic is intractable under $\mathcal{H}_0$ due to the overlapping nature of $\mathcal{W}$, a common approach, which will be considered here, is to use a Monte-Carlo method (see Section Computing the statistical significance of the MLC for more details).

Spatial scan statistics for multivariate data

Here we consider the case where several continuous variables are simultaneously observed in each spatial location: $X = (X^{(1)}, \ldots, X^{(p)})^\top$ is a p-dimensional variable ($p \geq 2$). In this context the objective is to identify multivariate spatial clusters that are aggregations of sites in which X takes higher or lower values (in terms of mean, median, etc.) than elsewhere. For example one could observe the average concentrations of several pollutants over a day: a vector can be associated with each site, each element of which corresponds to the average concentration of one pollutant. In this context a spatial cluster corresponds to a set of sites under or overexposed to multiple pollutants. Different approaches will be presented: a parametric method based on a Gaussian model and a nonparametric one.
Figure 2 summarizes the different types of multivariate data with examples, and provides guidelines on the spatial scan statistics methods to be used for these data (and the argument to use in the scan function of the package). More precisely, we distinguish three types of spatial data: lattice data which are aggregated data for example at the scale of the regions of a country, geostatistical data which are defined on a continuous space (typically temperature, sunshine, or atmospheric pressure) although they are observed only at discrete sites, and marked point data for which the location is random (for example the distribution of trees in a forest) and we observe at each location the circum-

ference and height of the tree for example. To detect spatial clusters, in the case of Gaussian data we will prefer the Gaussian approach (MG) and otherwise we will use the nonparametric approach (MNP).

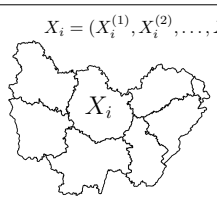
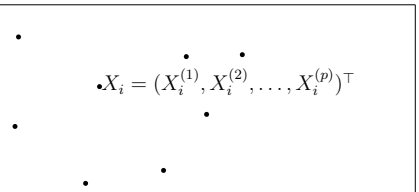
Data			
Application example	Lattice data: • Unemployment rate and fraction of the population that has not graduated from high school	Geostatistical data: • Temperature and air pressure	Point pattern: • Circumference and height of trees
Question	Is there a statistically significant cluster of high unemployment rates and high fraction of the population with a low level of education?	Is there a statistically significant cluster of high temperatures and low air pressure?	Is there a statistically significant cluster of trees with larger circumferences and heights?
Methods	Gaussian data: • Multivariate Gaussian spatial scan statistic • "MG" argument in the scan function of the package Non-Gaussian data: • Multivariate Nonparametric spatial scan statistic • "MNP" argument in the scan function of the package		
Interpretation	There is a statistically significant cluster and by describing the mean or median of each variable, we can get an indication of which variables are dominant in the cluster, and which variables are higher or lower in that cluster.		

Figure 2: Summary of spatial scan statistics for multivariate data. The table indicates the question that can be asked for the detection of clusters. It then indicates the methods to be used according to the distribution of the data as well as the ways to interpret the detected clusters. Spatial scan statistics for multivariate data can be used to detect spatial clusters on any type of spatial data (lattice, geostatistical, point data) modeled by vectors. The detected clusters can be characterized by computing the mean or median of each variable inside and outside each cluster.

Cucala et al. (2017) proposed a parametric spatial scan statistic for multivariate data based on a multivariate normal model taking into account the correlations between the variables.
The null hypothesis $\mathcal{H}_0$, corresponding to the absence of any cluster in the data, is the following: $\forall i \in [\![1;n]\!],\ X_i \sim \mathcal{N}_p(\mu, \Sigma)$ and the alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w can be defined as: $\forall i \in [\![1;n]\!],\ X_i \sim \begin{cases} \mathcal{N}_p(\mu_w, \Sigma_{w,w^c}) & \text{if } s_i \in w \\ \mathcal{N}_p(\mu_{w^c}, \Sigma_{w,w^c}) & \text{otherwise} \end{cases}$.

Then we can compute the MLE estimates of $\mu, \mu_w, \mu_{w^c}, \Sigma$ and Σ_{w,w^c}: $\hat{\mu}, \hat{\mu}_w, \hat{\mu}_{w^c}, \hat{\Sigma}$ and $\hat{\Sigma}_{w,w^c}$, and we can show that the log-likelihood ratio between these two hypotheses is

$$\widehat{LLR^w} = -\frac{n}{2}\ln\left[\det\Big(\sum_{\substack{i \\ s_i \in w}} (X_i - \hat{\mu}_w)(X_i - \hat{\mu}_w)^\top + \sum_{\substack{i \\ s_i \in w^c}} (X_i - \hat{\mu}_{w^c})(X_i - \hat{\mu}_{w^c})^\top\Big)\right] + \frac{n}{2}\ln\left[\det\Big(\sum_{i=1}^{n}(X_i - \hat{\mu})(X_i - \hat{\mu})^\top\Big)\right],$$

where $\hat{\mu}_g = \dfrac{1}{|g|}\displaystyle\sum_{i, s_i \in g} X_i$ for $g \in \{w, w^c\}$ and $\hat{\mu} = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} X_i$.

Finally the log-likelihood ratio is used as a concentration index and maximised over the set of potential clusters $\mathcal{W}$.

Thus we can show that the multivariate Gaussian (MG) scan statistic is

$$\lambda_{\text{MG}} = \min_{w \in \mathcal{W}} \det\Big(\sum_{\substack{i \\ s_i \in w}} (X_i - \widehat{\mu}_w)(X_i - \widehat{\mu}_w)^\top + \sum_{\substack{i \\ s_i \in w^c}} (X_i - \widehat{\mu}_{w^c})(X_i - \widehat{\mu}_{w^c})^\top \Big).$$

This test performs very well against Gaussian alternatives but faces problems when the data is not normal, which is often the case when dealing with environmental data exhibiting extreme values. For that reason Cucala et al. (2019) developed a nonparametric spatial scan statistic for multivariate data based on a multivariate extension of the Wilcoxon-Mann-Whitney test for multivariate data (Oja and Randles, 2004).
In this context the null hypothesis $\mathcal{H}_0$ can be rewritten as $\mathcal{H}_0 : X_1, \ldots, X_n$ are identically distributed, whatever the associated location.
Let

$$\begin{array}{rccl} \text{sgn} & : & \mathbb{R}^p & \to & \mathbb{R}^p \\ & & x & \mapsto & \begin{cases} ||x||_2^{-1} x & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases} \end{array},$$

then the multivariate ranks R_i are defined by $R_i = \frac{1}{n} \sum_{j=1}^{n} \text{sgn}(A_X(X_i - X_j))$ where the matrix A_X makes the ranks such that $\frac{p}{n} \sum_{i=1}^{n} R_i R_i^\top = \frac{1}{n} \sum_{i=1}^{n} R_i^\top R_i I_p$. Note that this matrix can be easily computed using an iterative procedure. Then the multivariate extension of the Wilcoxon-Mann-Whitney statistic proposed by Oja and Randles (2004) is

$$U^2(w) = \frac{p}{c_X^2} \left[|w| \, ||\bar{R}_w||_2^2 + |w^c| \, ||\bar{R}_{w^c}||_2^2 \right], \text{ where } c_X^2 = \frac{1}{n} \sum_{i=1}^{n} R_i^\top R_i.$$

Cucala et al. (2019) used $U^2(w)$ as a concentration index to build the spatial scan statistic: the multivariate nonparametric (MNP) scan statistic is $\lambda_{\text{MNP}} = \max_{w \in \mathcal{W}} U^2(w)$.

It should be noted that in the case $p = 1$, these statistics are respectively equivalent to the ones introduced by Kulldorff et al. (2009) (which is equivalent to the scan statistic developed by Cucala (2014), UG), and Jung and Cho (2015) (UNP).

Spatial scan statistics for univariate functional data

Here we consider the case where a continuous variable is observed in each spatial location over time: $\{X(t),\ t \in \mathcal{T}\}$ is a real-valued stochastic process where $\mathcal{T}$ is an interval of $\mathbb{R}$. In this context the objective is to identify functional spatial clusters that are aggregations of sites in which the curves are higher or lower than elsewhere. For example, one can observe the concentration of an air pollutant over time in different geographical areas. Then a cluster corresponds to an aggregation of sites in which the concentration of the air pollutant is higher or lower over the time than in the other spatial units. Several methods will be considered: a parametric method based on a functional ANOVA, a nonparametric approach using a Wilcoxon-Mann-Whitney test for high-dimensional data, a distribution-free approach based on a pointwise Student's t-test and finally a pointwise rank-based method. On Gaussian data, for non localized clusters in time all approaches show high power and high true positive rates. However the performances of the ANOVA-based method strongly decrease on non-normal data. For localized clusters in time (that are aggregations of sites that take higher or lower values for X only in a small interval of time (an interval of five days over a study period of one month for example)) the pointwise approaches should be favored.
Figure 3 summarizes the different types of univariate functional data with examples, and provides recommendations on the spatial scan statistics methods to be used for these data (and the argument to use in the scan function of the package). More precisely, we distinguish lattice functional data which are aggregated functional data for example at the scale of the administrative areas of a country (unemployment rate, percentage of the population over 65, etc), geostatistical functional data which are defined on a continuous space (typically temperature, sunshine, or atmospheric pressure over time) although they are observed only at discrete sites, and marked point data for which the location is random (for example the distribution of trees in a forest) and we observe at each location the circumference of the tree over time for example. To detect spatial clusters, as mentioned before, in the case of Gaussian data we will prefer the pointwise distribution-free functional approach (DFFSS) and otherwise we will use the pointwise rank-based approach (URBFSS).

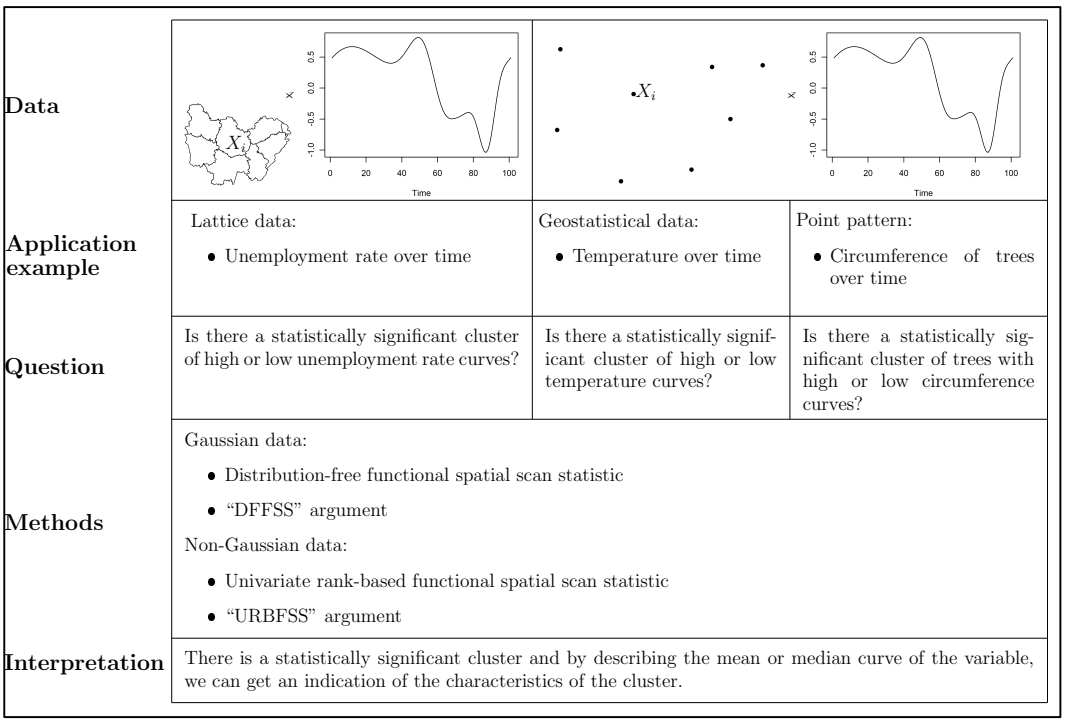


Figure 3: Summary of spatial scan statistics for univariate functional data. The table indicates the question that can be asked for the detection of clusters on a set of curves. It then indicates the methods to be used according to the distribution of the data as well as the ways to interpret the detected clusters. Spatial scan statistics for univariate functional data can be used to detect spatial clusters on any type of spatial data (lattice, geostatistical, point data) observed over a period of time. The detected clusters can be characterized by computing the mean or median curve inside and outside each cluster.

The parametric spatial scan statistic for univariate functional data

Frévent et al. (2021a) supposed that the process X takes values in a semi-metric space, in particular in $\mathcal{L}^2(\mathcal{T}, \mathbb{R})$ and proposed a parametric spatial scan statistic for functional data, based on a functional ANOVA. Here the null hypothesis $\mathcal{H}_0$ can be rewritten: $\mathcal{H}_0 : \forall w \in \mathcal{W},\ \mu_w = \mu_{w^c} = \mu_S$, and the alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w can be defined as follows: $\mathcal{H}_1^{(w)} : \mu_w \neq \mu_{w^c}$, where μ_w, μ_{w^c} and μ_S stand for the mean functions in w, outside w and over S, respectively. Cuevas et al. (2004) and Górecki and Smaga (2015) proposed the following ANOVA test statistic:

$$F_n^{(w)} = \frac{|w|\ ||\bar{X}_w - \bar{X}||_2^2 + |w^c|\ ||\bar{X}_{w^c} - \bar{X}||_2^2}{\frac{1}{n-2}\left[\sum_{j,s_j \in w} ||X_j - \bar{X}_w||_2^2 + \sum_{j,s_j \in w^c} ||X_j - \bar{X}_{w^c}||_2^2\right]},$$

where $\bar{X}_g(t) = \frac{1}{|g|} \sum_{i,s_i \in g} X_i(t)$ are empirical estimators of μ_g ($g \in \{w, w^c\}$), $\bar{X}(t) = \frac{1}{n} \sum_{i=1}^{n} X_i(t)$ is the empirical estimator of μ_S and $||x||_2^2 = \int_{\mathcal{T}} x^2(t)\ \mathrm{d}t$.

Thus, Frévent et al. (2021a) proposed to use $F_n^{(w)}$ as a concentration index and the proposed parametric functional spatial scan statistic (PFSS) is $\Lambda_{\text{PFSS}} = \max_{w \in \mathcal{W}} F_n^{(w)}$.

This method gives high powers and F-measures on normal data but as in the multivariate framework the parametric method faces problems when the data is not normal. Smida et al. (2022) proposed a nonparametric spatial scan statistic for functional data based on a functional Wilcoxon-Mann-Whitney test (Chakraborty and Chaudhuri, 2014).

A nonparametric spatial scan statistic for functional data

Here X is a process of a smooth Banach space χ, with a Gâteaux differentiable norm $||.||_\chi$. Let denote P_w and P_{w^c} the probability measures of X in w and in w^c respectively, then $\mathcal{H}_0$ corresponds to: $\mathcal{H}_0 : \forall w \in \mathcal{W},\ P_w = P_{w^c}$ and the alternative hypothesis associated with a potential cluster w can be rewritten as $\mathcal{H}_1^{(w)} : P_w(X) = P_{w^c}(X - \Delta),\ \Delta \in \chi \backslash \{0\}$.
Chakraborty and Chaudhuri (2014) defined the sign function in the functional framework as

$$\forall h \in \chi,\ \mathrm{Sgn}_X(h) = \begin{cases} \lim\limits_{v \to 0^+} \dfrac{||X + vh||_\chi - ||X||_\chi}{v} & \text{if } X \neq 0 \\ 0 & \text{if } X = 0 \end{cases}.$$

Then they proposed the following test statistic:

$$T_{WMW}(w) = \frac{1}{|w||w^c|} \sum_{i, s_i \in w} \sum_{j, s_j \in w^c} \mathrm{Sgn}_{X_j - X_i}.$$

Under $\mathcal{H}_0$, if $\dfrac{|w|}{n} \to \gamma \in [0; 1]$ as $|w|, |w^c| \to \infty$, $\sqrt{\dfrac{|w||w^c|}{n}} T_{WMW}(w)$ converges weakly to a distribution that does not depend on $|w|$. Thus Smida et al. (2022) proposed to use $U(w) = \left|\left| \sqrt{\dfrac{|w||w^c|}{n}} T_{WMW}(w) \right|\right|$ as a concentration index: the nonparametric functional scan statistic (NPFSS) is $\Lambda_{\mathrm{NPFSS}} = \max\limits_{w \in \mathcal{W}} U(w)$.
It should be noticed that although Smida et al. (2022) only studied the performances of the NPFSS in the univariate functional framework, their method is also applicable on multivariate functional data as shown by Frévent et al. (2021b).

A distribution-free spatial scan statistic for univariate functional data

Frévent et al. (2021a) also proposed to combine the distribution-free spatial scan statistic for univariate data proposed by Cucala (2014) and the max statistic of Lin et al. (2021). They supposed that for each time t, $\mathbb{V}[X_i(t)] = \sigma^2(t)$ for all $i \in [\![1; n]\!]$. Then for each t, the concentration index proposed by Cucala (2014) to test $\mathcal{H}_0 : \forall w \in \mathcal{W}, \mu_w(t) = \mu_{w^c}(t) = \mu_S(t)$ was

$$I^{(w)}(t) = \frac{|\bar{X}_w(t) - \bar{X}_{w^c}(t)|}{\sqrt{\hat{\mathbb{V}}[\bar{X}_w(t) - \bar{X}_{w^c}(t)]}},$$

where $\hat{\mathbb{V}}[\bar{X}_w(t) - \bar{X}_{w^c}(t)] = \widehat{\sigma^2}(t) \left[\dfrac{1}{|w|} + \dfrac{1}{|w^c|}\right]$,

$$\widehat{\sigma^2}(t) = \frac{1}{n-2} \left[\sum_{i, s_i \in w} (X_i(t) - \bar{X}_w(t))^2 + \sum_{i, s_i \in w^c} (X_i(t) - \bar{X}_{w^c}(t))^2 \right].$$

Then the idea is to globalize the information by maximizing the previous quantity over the time for each potential cluster w, as suggested by Lin et al. (2021):

$$I^{(w)} = \sup_{t \in \mathcal{T}} I^{(w)}(t).$$

For cluster detection, as for the PFSS, the null hypothesis $\mathcal{H}_0$ (the absence of cluster) is defined as follows: $\mathcal{H}_0 : \forall w \in \mathcal{W},\ \mu_w = \mu_{w^c} = \mu_S$. And the alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w can be defined as follows: $\mathcal{H}_1^{(w)} : \mu_w \neq \mu_{w^c}$.

Frévent et al. (2021a) considered $I^{(w)}$ as a concentration index and maximized it over the set of potential clusters $\mathcal{W}$ yielding to the following distribution-free functional spatial scan statistic (DFFSS): $\Lambda_{\mathrm{DFFSS}} = \max\limits_{w \in \mathcal{W}} I^{(w)}$.

A new rank-based spatial scan statistic for univariate functional data

A pointwise approach based on ranks and on the nonparametric scan statistic for univariate data (Jung and Cho, 2015) can be proposed in the univariate functional framework by adapting the approach of Frévent et al. (2021b).

For a time t, Jung and Cho (2015) proposed to test $\mathcal{H}_0 : \forall w \in \mathcal{W}, F_{w,t} = F_{w^c,t}$ where $F_{w,t}$ and $F_{w^c,t}$ are the cumulative distribution functions of $X(t)$ in w and outside w, by using the Wilcoxon rank-sum test statistic. For a time t and a potential cluster w, the Wilcoxon rank-sum test statistic is $W(t)^{(w)} = \sum_{i,s_i \in w} R_i(t)$ where $R_i(t)$ is the rank of $X_i(t)$ in $\{X_1(t), \ldots, X_n(t)\}$, using the average rank in the case of tied observations.
Then the standardized version of this statistic is

$$T(t)^{(w)} = \frac{W(t)^{(w)} - \mathbb{E}[W(t)^{(w)}]}{\sqrt{\mathbb{V}[W(t)^{(w)}]}}$$

where $\mathbb{E}[W(t)^{(w)}] = \frac{|w|(n+1)}{2}$ and $\mathbb{V}[W(t)^{(w)}] = \frac{|w||w^c|(n+1)}{12}$ are the expected value and the variance of $W(t)^{(w)}$ under $\mathcal{H}_0$.
Jung and Cho (2015) proposed to minimize the p-value associated with $T(t)^{(w)}$ on the set of potential clusters $\mathcal{W}$. We propose to adapt their approach by simply using $|T(t)^{(w)}|$ as a pointwise statistic.

In the context of cluster detection, the null hypothesis is defined as $\mathcal{H}_0$: $\forall w \in \mathcal{W}$, $\forall t \in \mathcal{T}$, $F_{w,t} = F_{w^c,t}$. The alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w is $\mathcal{H}_1^{(w)}$: $\exists t \in \mathcal{T}$, $F_{w,t}(x) = F_{w^c,t}(x - \Delta_t)$, $\Delta_t \neq 0$.

As before, we propose to globalize the information over the time with $T^{(w)} = \sup_{t \in \mathcal{T}} |T(t)^{(w)}|$ and to use this quantity as a concentration index, yielding to the following univariate rank-based functional spatial scan statistic (URBFSS): $\Lambda_{\text{URBFSS}} = \max_{w \in \mathcal{W}} T^{(w)}$.

Spatial scan statistics for multivariate functional data

Here we consider the case where several continuous variables are observed simultaneously in each spatial unit over time: $\{X(t),\ t \in \mathcal{T}\}$ is a p-dimensional vector-valued stochastic process ($p \geq 2$) where $\mathcal{T}$ is an interval of $\mathbb{R}$. The objective is to detect multivariate functional spatial clusters that are aggregations of sites in which the curves are higher or lower than elsewhere. For example we can observe the concentration of several pollutants over time in different locations. Thus at each location we observe several processes (air pollutant concentrations) and these processes can be correlated. In this context a cluster is an aggregation of sites overexposed or underexposed to multiple pollutants over time. Several methods will be presented: a parametric method based on a functional MANOVA, a distribution-free approach based on a pointwise Hotelling T^2-test and finally a pointwise rank-based method. On normal data, all approaches show high power and high true positive rates for non localized clusters in time. However the performances of the methods based on the MANOVA and the Hotelling T^2-test decrease on non-normal data. For localized clusters in time the pointwise approaches should be favored, especially the pointwise rank-based method on non-Gaussian data. By localized clusters in time we mean aggregations of sites that take higher or lower values for X only in a small interval of time (an interval of five days over a study period of one month for example).
Figure 4 summarizes the different types of multivariate functional data with examples, and provides guidelines on the spatial scan statistics methods to be used for these data (and the argument to use in the scan function of the package). To be more precise, we can distinguish lattice functional data which are aggregated data for example at the scale of the regions of a country (unemployment rate and fraction of the population that has not graduated from high school, over time, for example), geostatistical functional data which are defined on a continuous space (temperature, sunshine, and atmospheric pressure over time) although they are observed only at discrete sites, and marked point data for which the location is random (for example the distribution of trees in a forest) and we observe at each location the circumference and height of the tree over time for example. As previously mentioned, to detect spatial clusters, in the case of Gaussian data we will prefer the multivariate distribution-free functional spatial scan statistic (MDFFSS) and for non-Gaussian data we will use the pointwise rank-based approach (MRBFSS).

A parametric spatial scan statistic for multivariate functional data

Here, the process X is supposed to take values in a semi-metric space, in particular the Hilbert space $\mathcal{L}^2(\mathcal{T}, \mathbb{R}^p)$ of p-dimensional vector-valued square-integrable functions on $\mathcal{T}$, equipped with the inner

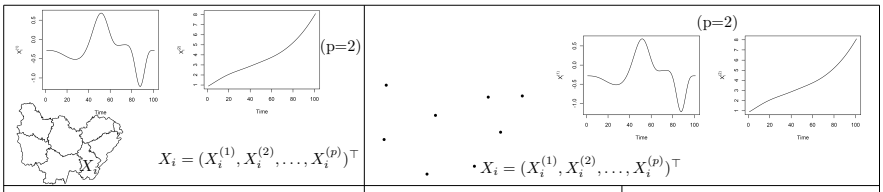
<table>
<tr><td>Data</td><td colspan="3"> 
</td></tr>
<tr><td>Application example</td><td>Lattice data:
• Unemployment rate and fraction of the population that has not graduated from high school over time</td><td>Geostatistical data:
• Temperature and air pressure over time</td><td>Point pattern:
• Circumference and height of trees over time</td></tr>
<tr><td>Question</td><td>Is there a statistically significant cluster of high unemployment rate curves and high fraction of the population with a low level of education over time?</td><td>Is there a statistically significant cluster of high temperature and low air pressure curves?</td><td>Is there a statistically significant cluster of trees with high circumference and height curves?</td></tr>
<tr><td>Methods</td><td colspan="3">Gaussian data:
• Multivariate distribution-free functional spatial scan statistic
• “MDFFSS” argument in the scan function of the package
Non-Gaussian data:
• Multivariate rank-based functional spatial scan statistic
• “MRBFSS” argument in the scan function of the package</td></tr>
<tr><td>Interpretation</td><td colspan="3">There is a statistically significant cluster and by describing the mean or median curve of each variable, we can get an indication of which variables are dominant in the cluster, and which variables present higher or lower curves in that cluster.</td></tr>
</table>

Figure 4: Summary of spatial scan statistics for multivariate functional data. The table indicates the question that can be asked for the detection of clusters of multivariate curves. It then indicates the methods to be used according to the distribution of the data as well as the ways to interpret the detected clusters. Spatial scan statistics for multivariate functional data can be used to detect spatial clusters on any type of spatial data (lattice, geostatistical, point data) composed of multivariate curves X_i (in the table example, $X_i = (X_i^{(1)}, X_i^{(2)})^\top$ is composed of two curves). Then, the detected clusters can be characterized by computing the mean or median curve of each variable inside and outside each cluster.

product $\langle X, Y \rangle = \int_{\mathcal{T}} X(t)^\top Y(t)\, \mathrm{d}t$.

Frévent et al. (2021b) proposed a parametric scan statistic for multivariate functional data based on a functional MANOVA Lawley–Hotelling trace test (Górecki and Smaga, 2017).

In this context, the null hypothesis $\mathcal{H}_0$ is $\mathcal{H}_0 : \forall w \in \mathcal{W},\ \mu_w = \mu_{w^c} = \mu_S$, where μ_w, μ_{w^c} and μ_S stand for the mean functions in w, outside w and over S, respectively. And the alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w is $\mathcal{H}_1^{(w)} : \mu_w \neq \mu_{w^c}$. Górecki and Smaga (2017) presented the following adaptation of the Lawley-Hotelling trace test statistic:

$$\mathrm{LH}^{(w)} = \mathrm{Trace}(H_w E_w^{-1})$$

where $H_w = |w| \int_{\mathcal{T}} [\bar{X}_w(t) - \bar{X}(t)]\, [\bar{X}_w(t) - \bar{X}(t)]^\top \, \mathrm{d}t + |w^c| \int_{\mathcal{T}} [\bar{X}_{w^c}(t) - \bar{X}(t)]\, [\bar{X}_{w^c}(t) - \bar{X}(t)]^\top \, \mathrm{d}t$

and $E_w = \sum_{j, s_j \in w} \int_{\mathcal{T}} \left[X_j(t) - \bar{X}_w(t)\right] \left[X_j(t) - \bar{X}_w(t)\right]^\top \mathrm{d}t + \sum_{j, s_j \in w^c} \int_{\mathcal{T}} \left[X_j(t) - \bar{X}_{w^c}(t)\right] \left[X_j(t) - \bar{X}_{w^c}(t)\right]^\top \mathrm{d}t$

with $\bar{X}_g(t) = \dfrac{1}{|g|} \sum_{i, s_i \in g} X_i(t)$ the empirical estimators of $\mu_g(t)$ for $g \in \{w, w^c\}$ and $\bar{X}(t) = \dfrac{1}{n} \sum_{i=1}^{n} X_i(t)$ the empirical estimator of $\mu_S(t)$.

Then Frévent et al. (2021b) considered $\mathrm{LH}^{(w)}$ as a concentration index and proposed the parametric multivariate functional spatial scan statistic (MPFSS): $\Lambda_{\mathrm{MPFSS}} = \max_{w \in \mathcal{W}} \mathrm{LH}^{(w)}$.

In fact Górecki and Smaga (2017) proposed four test statistics using the matrices H_w and E_w to compare the mean functions in w and w^c: (1) the Lawley–Hotelling trace test statistic $\mathrm{LH}^{(w)} = \mathrm{Trace}(H_w E_w^{-1})$, (2) the Pillai trace test statistic $\mathrm{P}^{(w)} = \mathrm{Trace}(H_w (H_w + E_w)^{-1})$, (3) the Roy's largest root test statistic $\mathrm{R}^{(w)} = \lambda_{\max}(H_w E_w^{-1})$ where $\lambda_{\max}(H_w E_w^{-1})$ is the maximum eigenvalue of $H_w E_w^{-1}$ and (4) the Wilks

lambda test statistic $\mathrm{W}^{(w)} = \dfrac{\det(E_w)}{\det(H_w + E_w)}$.

Thus each of these quantities (or the opposite for the Wilks lambda test statistic) can be considered as a concentration index and maximized over $\mathcal{W}$ which results in the following parametric multivariate functional spatial scan statistics:

$$\Lambda_{\mathrm{LH}} = \max_{w\in\mathcal{W}} \mathrm{LH}^{(w)},\ \Lambda_{\mathrm{P}} = \max_{w\in\mathcal{W}} \mathrm{P}^{(w)},\ \Lambda_{\mathrm{R}} = \max_{w\in\mathcal{W}} \mathrm{R}^{(w)},\ \Lambda_{\mathrm{W}} = \min_{w\in\mathcal{W}} \mathrm{W}^{(w)}.$$

These four approaches are implemented in the package **HDSpatialScan**.

A distribution-free spatial scan statistic for multivariate functional data

Frévent et al. (2021b) proposed a distribution-free spatial scan statistic for multivariate functional data which is the counterpart of the distribution-free spatial scan statistic for univariate functional data developed by Frévent et al. (2021a). They supposed that for each time t, $\mathbb{V}[X_i(t)] = \Sigma(t,t)$ for all $i \in [\![1;n]\!]$, where Σ is a $p \times p$ covariance matrix function.

Thus, as previously, in the context of cluster detection, the null hypothesis $\mathcal{H}_0$ can be defined as follows: $\mathcal{H}_0 : \forall w \in \mathcal{W},\ \mu_w = \mu_{w^c} = \mu_S$, where μ_w, μ_{w^c} and μ_S stand for the mean functions in w, outside w and over S, respectively. And the alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w can be defined as follows: $\mathcal{H}_1^{(w)} : \mu_w \neq \mu_{w^c}$. Next, Qiu et al. (2021) proposed to compare the mean function μ_w in w with the mean function μ_{w^c} in w^c by using the following statistic:

$$T_{n,\max}^{(w)} = \sup_{t\in\mathcal{T}} T_n(t)^{(w)}$$

where $T_n(t)$ is a pointwise statistic defined by the Hotelling T^2-test statistic

$$T_n(t)^{(w)} = \frac{|w||w^c|}{n} \left(\bar{X}_w(t) - \bar{X}_{w^c}(t)\right)^\top \hat{\Sigma}(t,t)^{-1} \left(\bar{X}_w(t) - \bar{X}_{w^c}(t)\right).$$

$\bar{X}_w(t)$ and $\bar{X}_{w^c}(t)$ are the empirical estimators of the mean functions defined previously, and

$$\hat{\Sigma}(s,t) = \frac{1}{n-2}\left[\sum_{i,s_i\in w} \left(X_i(s) - \bar{X}_w(s)\right)\left(X_i(t) - \bar{X}_w(t)\right)^\top + \sum_{i,s_i\in w^c} \left(X_i(s) - \bar{X}_{w^c}(s)\right)\left(X_i(t) - \bar{X}_{w^c}(t)\right)^\top\right]$$

is the pooled sample covariance matrix function.

Then Frévent et al. (2021b) proposed to use $T_{n,\max}^{(w)}$ as a concentration index and to maximize it over the set of potential clusters $\mathcal{W}$: the multivariate distribution-free functional spatial scan statistic (MDFFSS) is $\Lambda_{\mathrm{MDFFSS}} = \max_{w\in\mathcal{W}} T_{n,\max}^{(w)}$.

A rank-based spatial scan statistic for multivariate functional data

Finally Frévent et al. (2021b) also proposed to consider as a pointwise test statistic the multivariate extension of the Wilcoxon rank-sum test statistic developed by Oja and Randles (2004) and detailed in Section Spatial scan statistics for multivariate data. They defined the pointwise multivariate ranks as $R_i(t) = \dfrac{1}{n}\sum_{j=1}^{n} \mathrm{sgn}(A_X(t)(X_i(t) - X_j(t)))$ where the pointwise transformation matrix $A_X(t)$ is so that $\dfrac{p}{n}\sum_{i=1}^{n} R_i(t)R_i(t)^\top = \dfrac{1}{n}\sum_{i=1}^{n} R_i(t)^\top R_i(t) I_p$, and the sgn function is the same as in Section Spatial scan statistics for multivariate data.

Then for each time t, the pointwise multivariate extension of the Wilcoxon rank-sum test statistic is defined as $W(t)^{(w)} = \dfrac{pn}{\sum_{i=1}^{n} R_i(t)^\top R_i(t)}\left[\,|w|\,||\bar{R}_w(t)||_2^2 + |w^c|\,||\bar{R}_{w^c}(t)||_2^2\,\right]$ where

$$\bar{R}_g(t) = \frac{1}{|g|}\sum_{i,s_i\in g} R_i(t)\ (g \in \{w, w^c\}).$$

In the context of cluster detection, the null hypothesis is defined as $\mathcal{H}_0$: $\forall w \in \mathcal{W},\ \forall t \in \mathcal{T},\ F_{w,t} = F_{w^c,t}$ where $F_{w,t}$ and $F_{w^c,t}$ correspond respectively to the cumulative distribution functions of $X(t)$ in w and outside w. The alternative hypothesis $\mathcal{H}_1^{(w)}$ associated with a potential cluster w is $\mathcal{H}_1^{(w)}$:

$\exists t \in \mathcal{T},\ F_{w,t}(x) = F_{w^c,t}(x - \Delta_t), \Delta_t \neq 0.$

Finally Frévent et al. (2021b) proposed to globalize the information over the time with the quantity $W^{(w)} = \sup_{t \in \mathcal{T}} W(t)^{(w)}$ and to use it as a concentration index to be maximized over the set of potential clusters $\mathcal{W}$. The multivariate rank-based functional spatial scan statistic (MRBFSS) is then $\Lambda_{\text{MRBFSS}} = \max_{w \in \mathcal{W}} W^{(w)}$.

Computing the statistical significance of the MLC

Once the most likely cluster (MLC) is detected, its statistical significance must be evaluated. The distribution of the scan statistic $\mathcal{S}$ ($\mathcal{S} = \lambda_{\text{MG}}$, λ_{MNP}, Λ_{PFSS}, Λ_{NPFSS}, Λ_{DFFSS}, Λ_{URBFSS}, Λ_{MPFSS}, Λ_{MDFFSS} or Λ_{MRBFSS}) is intractable under $\mathcal{H}_0$ due to the overlapping nature of $\mathcal{W}$. Then we choose to obtain a large set of simulated datasets by randomly permuting the observations X_i in the spatial locations. This technique was already used in spatial scan statistics (Kulldorff et al., 2009; Cucala et al., 2017).

Let M denote the number of random permutations of the original dataset and $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(M)}$ be the observed scan statistics on the simulated datasets. According to Dwass (1957) the p-value for $\mathcal{S}$ observed in the real data is estimated by

$$\hat{p} = \frac{1 + \sum_{m=1}^{M} \mathbb{1}_{\mathcal{S}^{(m)} \geq \mathcal{S}}}{M + 1}. \tag{2}$$

Finally, the MLC is considered to be statistically significant if the associated $\hat{p}$ is less than the type I error.

How to choose the method to apply to the data?

According to Cucala et al. (2019) the MNP method tends to present a better power and higher true positive rates for non-Gaussian data than the MG one. Although the false positive rates are often higher for this approach than the MG one, it remains moderate. The same conclusions are true for the UG (Kulldorff et al., 2009) and the UNP (Jung and Cho, 2015) which are their particular counterparts in the case of a single variable. In the functional framework, the approaches that present the best results are the DFFSS and the URBFSS in the univariate context and the MDFFSS and the MRBFSS in the multivariate one (Frévent et al., 2021a,b). The URBFSS and the MRBFSS tend to show higher powers and higher true positive rates although they detect more false positives than the DFFSS and the MDFFSS respectively. Table 1 summarizes the methods and their performances. The symbols ✓ and X indicate respectively a high and a low performance on the criterion. If there is no symbol it means that for this criterion the approach offers medium performances. The terminology "localized clusters in time" in the functional cases refers to aggregations of sites that take higher or lower values for the process only in a small interval of time (an interval of five days over a study period of one month for example). Table 2 gives an idea of the computation time of the different scanning methods proposed by the package. It should be noted that the computation time of the different spatial scan statistics methods is dependent on the size of the datasets, and in particular a function of the number of sites, the number of observation times and the number of variables considered.

3 Software

Computing the spatial scan statistic

The package **HDSpatialScan** provides a function `SpatialScan` to compute all the spatial scan statistics. The user chooses the method to apply by specifying the `method` argument: `"MG"` and `"MNP"` apply respectively the parametric and nonparametric spatial scan statistics approaches on multivariate data. Their univariate counterparts (when $p = 1$) can be computed with `"UG"` and `"UNP"` respectively. Then `"PFSS"`, `"DFFSS"` and `"URBFSS"` apply the parametric, the distribution-free and the new rank-based functional approaches on univariate functional data, and `"MPFSS"`, `"MDFFSS"`, and `"MRBFSS"` are their multivariate counterparts. Finally `"NPFSS"` applies the nonparametric spatial scan statistic for functional data developed by Smida et al. (2022) on both univariate and multivariate functional data.

Table 1: Performance in terms of power, true positive rate and false positive rate of spatial scan statistics for multivariate data (MG and MNP), univariate functional data (PFSS, DFFSS, NPFSS and URBFSS) and multivariate functional data (MPFSS, MDFFSS, NPFSS and MRBFSS)

	Gaussian distribution			**Non-Gaussian distribution**		
Method	Power	True positive rate	False positive rate	Power	True positive rate	False positive rate
		Univariate data			Univariate data	
UG[a]	✓	✓	✓	X	X	✓
UNP[a]	✓	✓		✓	✓	
		Multivariate data			Multivariate data	
MG[b]	✓	✓	✓	X	X	✓
MNP[b]	✓	✓		✓	✓	
		Functional univariate data			Functional univariate data	
		Non localized clusters in time			Non localized clusters in time	
PFSS[c]			✓	X		✓
DFFSS[d]	✓	✓	✓	✓	✓	✓
NPFSS[e]		✓			✓	
URBFSS[f]	✓	✓		✓	✓	
		Localized clusters in time			Localized clusters in time	
PFSS[c]	X	X		X	X	
DFFSS[d]	✓	✓	✓	✓	✓	✓
NPFSS[e]	X			X		
URBFSS[f]	✓	✓	✓	✓	✓	✓
		Functional multivariate data			Functional multivariate data	
		Non localized clusters in time			Non localized clusters in time	
MPFSS[c]			✓	X	X	✓
MDFFSS[d]	✓	✓	✓			✓
NPFSS[e]		✓		✓	✓	
MRBFSS[f]	✓	✓		✓	✓	
		Localized clusters in time			Localized clusters in time	
MPFSS[c]	X	X		X	X	✓
MDFFSS[d]	✓	✓	✓			✓
NPFSS[e]	X			X		
MRBFSS[f]	✓	✓	✓	✓	✓	✓

[a] The Univariate Gaussian (UG) and the Univariate Nonparametric (UNP) spatial scan statistics
[b] The Multivariate Gaussian (MG) and the Multivariate Nonparametric (MNP) spatial scan statistics
[c] The Parametric Functional (PFSS) and the Multivariate Parametric Functional (MPFSS) spatial scan statistics
[d] The Distribution-free Functional (DFFSS) and the Multivariate Distribution-free Functional (MDFFSS) spatial Scan Statistics
[e] The Nonparametric Functional (NPFSS) spatial Scan Statistic
[f] The Univariate Rank-based Functional (URBFSS) and the Multivariate Rank-based Functional (MRBFSS) spatial Scan Statistics

Table 2: Estimation of the computation time (over 100 repetitions) for the different scan statistics methods among 169 sites with *a priori* clusters comprising between 1 and 50% of the sites (default parameters) and 99 permutations for the estimation of the associated p-values (the default parameter of 999 permutations multiplies the computation time by about 10). Parallelization by running seven tasks in parallel was used (except for UG and UNP since these two methods are optimized to have a very low computation time without using CPUs) on two hexacores of type Intel(R) Xeon(R) CPU E5-2620 v2. For multivariate data (functional or not) 4 variables are considered and for functional data (univariate or multivariate) 56 observation times are considered.

Method	**Computation time (in s)**			
	Mean	Standard deviation	Minimum	Maximum
Univariate data				
UG[a]	4.31	0.18	4.01	4.77
UNP[a]	3.07	0.12	2.77	3.5
Multivariate data				
MG[b]	253.04	32.93	231.34	310.91
MNP[b]	14.6	1.44	13.48	17.77
Functional univariate data				
PFSS[c]	113.51	8.08	109.47	132.58
DFFSS[d]	55.99	4.93	52.52	66.76
NPFSS[e]	12.91	1.23	11.74	15.6
URBFSS[f]	17.93	1.37	16.93	22.53
Functional multivariate data				
MPFSS[c]	72.52	9.21	65.23	100.17
MDFFSS[d]	182.95	21.54	166.27	227.84
NPFSS[e]	22.95	1.7	21.77	28
MRBFSS[f]	244.04	21.83	234.05	305.56

[a] The Univariate Gaussian (UG) and the Univariate Nonparametric (UNP) spatial scan statistics
[b] The Multivariate Gaussian (MG) and the Multivariate Nonparametric (MNP) spatial scan statistics
[c] The Parametric Functional (PFSS) and the Multivariate Parametric Functional (MPFSS) spatial scan statistics
[d] The Distribution-free Functional (DFFSS) and the Multivariate Distribution-free Functional (MDFFSS) spatial Scan Statistics
[e] The Nonparametric Functional (NPFSS) spatial Scan Statistic
[f] The Univariate Rank-based Functional (URBFSS) and the Multivariate Rank-based Functional (MRBFSS) spatial Scan Statistics

Type of the data

Depending on the type of approach (univariate, multivariate, functional univariate or functional multivariate), the data must be formatted in a specific way. For univariate approaches, the data must be a vector in which each element corresponds to a site. If the data is individual and many individuals share the same site, the data can remain in an individual format with one element of the vector per individual. Then for real-valued multivariate methods or functional univariate methods, the data must be a matrix in which each row corresponds to a site (or an individual) and each column corresponds to a variable or an observation time in the functional framework. For multivariate functional methods the data must be a list in which each element is a matrix corresponding to a site (or an individual). In the matrices, the rows correspond to the variables and the columns to the observation times. Note that the observation times must be the same for each site or individual and they must be equally spaced for the methods `"NPFSS"`, `"PFSS"` and `"MPFSS"`. However if it is not the case of the raw data, they can be easily transformed by smoothing the data (Ramsay and Silverman, 2005), by using for example the R package **fda** (Ramsay et al., 2020).

Parameters of the scan function

The most important parameter is the `method` argument which has already been presented previously and allows to choose the spatial scan statistics to be applied. Note that you can choose one or more methods. Supplying `"MPFSS"` automatically computes the four strategies for the multivariate parametric functional spatial scan statistic (the Lawley-Hotelling trace (LH), the Roy's largest root (R), the Pillai's trace (P) and the Wilks' lambda (W)). If you only want the Lawley-Hotelling trace for example, you can simply supply `"MPFSS-LH"`. Although the Lawley-Hotelling trace test is the most used statistic (Oja and Randles, 2004), it should be noted that all these methods usually provide very similar results. The other arguments are `data`, `sites_coord`, `system`, `mini`, `maxi`, `type_minimaxi`, `mini_post`, `maxi_post`, `type_minimaxi_post`, `sites_areas`, `MC`, `typeI`, `nbCPU`, `variable_names` and `times`. Note that `nbCPU` will be ignored for the methods `"UG"` and `"UNP"`, `variable_names` is ignored for the univariate and univariate functional scan statistics and `times` is ignored for non-functional scan statistics.

The argument `data`, is the data vector, matrix or list on which the approaches must be applied. `MC` and `typeI` correspond respectively to the number of permutations of the data while computing the statistical significance of the clusters and the type I error i.e. a cluster is declared significant if its estimated p-value is below this threshold.
The arguments `sites_coord` and `system` are respectively a matrix of two columns corresponding to the coordinates of each site or individual, and to the system of coordinates ("Euclidean" or "WGS84").
The `sites_areas` argument is optional and corresponds to the areas of the sites (or the site of each individual if the data is individual).
The argument `nbCPU` permits to do parallelization and the arguments `mini`, `maxi`, `type_minimaxi`, `mini_post`, `maxi_post`, `type_minimaxi_post` are described further below.
`variable_names` is simply the names of the variables (in the same order as in the data) for multivariate or multivariate functional scan statistics and `times` corresponds to the times of observation, they must be numeric.

***A priori* filtering** The clusters are computed automatically as circular clusters, so we need to define a minimum and a maximum size for these clusters. That is what we call "*a priori* filtering" and this allows to control the computation time. Three types of *a priori* filtering are possible through the argument `type_minimaxi`: `"sites/indiv"` (the filtering is applied on the number of sites or individuals in the potential clusters, it is the default value), `"area"` (it is applied on the area of the clusters and is available only if `sites_areas` is provided), or `"radius"` (the radius of the clusters).
The arguments `mini` and `maxi` are then respectively the minimum number of sites/individuals, or the minimal area or radius and the maximum number of sites/individuals, or the maximal area or radius. For the radius it is specified in km if `system` is `"WGS84"` or in the user units if `system` is `"Euclidean"`.
It should be noted that this filtering can bias the p-values obtained for the clusters. In order to perform a correct statistical inference, Kulldorff and Nagarwalla (1995) recommended to consider a maximum size of half the study region. Thus the default setting is to consider potential clusters comprising at least one site and at most 50% of the sites (Equation 1). If you want to select clusters according to size (number of sites or individuals), area or radius, it is better to select them *a posteriori* among the detected clusters and if you really want to decrease the computation time we recommend to increase the number of CPU (with the argument `nb_CPU`). Changing the default settings can allow the user to investigate whether there appear to be clusters in a relatively quick first step, although the inference is biased, before applying the scan procedure with the default settings for the *a priori* filtering (50% of the studied region).

***A posteriori* filtering** Sometimes after that the p-value of each potential cluster has been computing, the user may want to retrieve only the significant clusters that satisfy a certain size, area, or radius criteria. That is what we call *a posteriori* filtering. The corresponding arguments are `mini_post`, `maxi_post` and `type_minimaxi_post` and their definitions are the same as `mini`, `maxi` and `type_minimaxi`. If the user only wants to obtain clusters meeting size criteria, this *a posteriori* approach must be prioritized over the *a priori* approach which gives biased results and must therefore be used with great care.

Output of the scan function

The function `SpatialScan` returns a list of object of class `"ResScanOutput"` which is composed of many elements. The element `sites_clusters` is a list in which each element corresponds to a significant cluster and contains the index of the sites (or the individuals) included in this cluster. The clusters are listed in their order of detection. The secondary clusters are defined according to Kulldorff (1997): they correspond to potential clusters that also present large values for the concentration index. Their p-values are calculated as if they were the most likely cluster themselves which is a bit conservative since the secondary clusters are compared with the most likely cluster of the permutations (Kulldorff, 1997). Finally, only clusters that are significant at the `typeI` threshold and that do not overlap with a more likely cluster are returned, and `pval_clusters` corresponds to the associated p-values. The element `centres_clusters` corresponds to the coordinates of the centres of each detected cluster and `radius_clusters` is the radius of the clusters in km if `system` is "WGS84" or in the user units otherwise. `areas_clusters` corresponds to the areas of the clusters (in the same units as `sites_areas`). Finally the system of coordinates, the coordinates of the sites, the data and the name of the scan procedure are recalled respectively in the elements `system`, `sites_coord`, `data` and `method`.

Depending on the type of the method (univariate, multivariate, univariate functional or multivariate functional) the objects of class `"ResScanOutput"` are also of class `"ResScanOutputUni"`, `"ResScanOutputMulti"`, `"ResScanOutputUniFunct"` or `"ResScanOutputMultiFunct"`. The objects of class `"ResScanOutputMulti"` and `"ResScanOutputMultiFunct"` also include the element `variable_names`, and the objects of class `"ResScanOutputUniFunct"` and `"ResScanOutputMultiFunct"` include the element `time`.

Plot or summarize the results

It is possible to plot the detected clusters by using the classical `plot` function. Depending on the `type` parameter, the package **HDSpatialScan** provides three different types of plot.
The first one, `"map"`, allows the user to plot a map of the sites and draws the circles corresponding to the circular clusters. The second one, `"map2"`, plots the clusters in colors. For these two types of plot the argument `spobject` which is the spatial object corresponding to the sites, must be provided. If you do not have this object you can use the third type `"schema"` which simply draws a schema of the sites and the clusters, with the argument `system_conv` which allows to correctly project the coordinates. It must be entered as in the PROJ documentation (PROJ contributors, 2021).

One may also want to get some features of one's clusters.
The function `summary` allows to get a summary of the clusters, either the mean and the standard deviation of each of the variables (if many) if the argument `type_summ` is `"param"`, or the 25th percentiles, the medians and the 75th percentiles if the argument `type_summ` is `"nparam"`. This function also provides the p-values, the radius and the area if available (only if `sites_areas` is provided) for each cluster detected.
Other interesting functions are `plotCurves` that allows to display cluster curves (only in the functional case), and `plotSummary` which displays the average (if `type = "mean"`) or the median (if `type = "median"`) curves in the clusters, outside and the global mean or median curves in the functional case. For the multivariate non-functional framework it displays a spider chart of means or medians for each variable inside the cluster, outside, or in all the area. Note that all these functions take an argument `only.MLC` which allows to only plot or summarize the most likely cluster (by setting `only.MLC = TRUE`). Finally the `print` function shows the scan procedure used as well as the number of clusters detected and their p-value.

4 Illustrations

To show the simplicity of use of the package, we will apply the different approaches on the environmental data provided in the package. It should be noted that the codes presented in this section represent a total computation time of about one hour on a regular laptop, using 7 cores.

Air pollution in northern France

We considered data provided by the French national air quality forecasting platform PREV'AIR which is available in the package **HDSpatialScan**. This lattice data consists in the daily concentrations (from May 1, 2020 to June 25, 2020) in $\mu g.m^{-3}$ of four pollutants for each of the 169 *cantons* (administrative subdivisions of France) of the *Nord-Pas-de-Calais* (a region in northern France) characterized by spatial polygons and located by their center of gravity $s_1, \ldots, s_{169}$: nitrogen dioxide (NO_2), ozone (O_3) and fine particles PM_{10} and $PM_{2.5}$ corresponding respectively to particles whose diameter is less than $10\mu m$ and $2.5\mu m$. The package **HDSpatialScan** provides the full data: `fmulti_data` but also some reduced data for the univariate functional case which consists in considering only the NO_2 concentrations (`funi_data`), and for the multivariate non-functional framework (`multi_data`) which corresponds to the temporal mean concentrations of the four pollutants over the study period.

The first step is to load the data:

```
library(HDSpatialScan)
data("map_sites")
data("multi_data")
data("funi_data")
data("fmulti_data")
```

The second step is to visualize the pollutants daily concentration curves in each *canton* and the spatial distributions of the temporal mean concentrations for each pollutant over the studied time period (Figures 5 and 6). This step allows us to see if sites seem to aggregate and therefore if launching a cluster detection is relevant, and if a temporal variation of the concentrations is visible, in which case a functional method will be more relevant than a multivariate approach summarizing each curve by its mean.

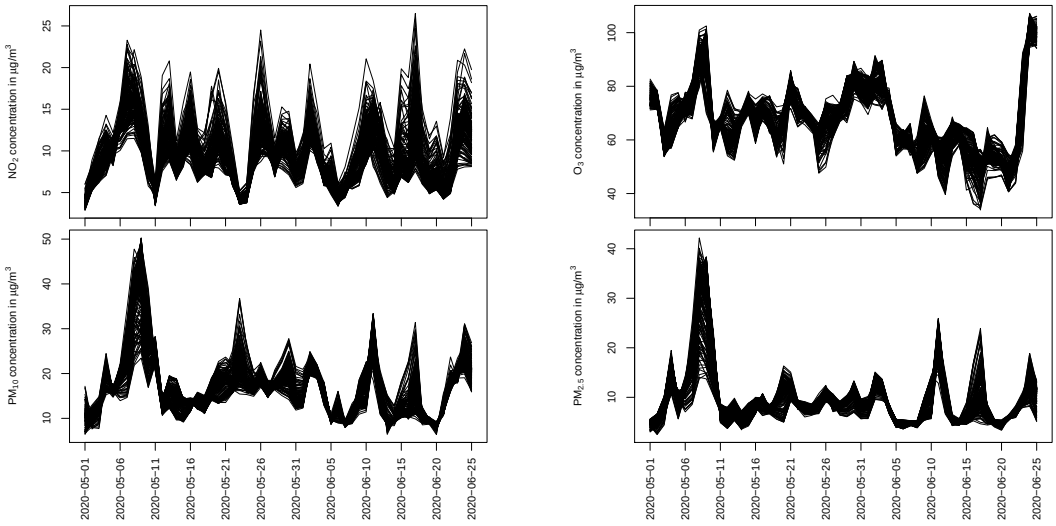


Figure 5: Daily concentration curves of NO_2, O_3, PM_{10} and $PM_{2.5}$ (from May 1, 2020 to June 25, 2020) in each of the 169 *cantons* of *Nord-Pas-de-Calais* (a region in northern France). A marked temporal variability in the concentrations of the four pollutants is observed over the study period.

The maps in Figure 6 show a spatial heterogeneity of the average concentration for each pollutant. Thus spatial scan statistics seem to be suitable to highlight the presence of *cantons*-level spatial clusters of pollutants concentrations. Moreover since the curves in Figure 5 show a marked temporal variability during the period from May 1, 2020 to June 25, 2020 a functional approach is more appropriate. However for sake of completeness we will also perform a multivariate spatial scan statistic approach anyway. Since small clusters of pollution are more relevant for interpretation because the sources of the pollutants are very localized, we will consider an *a posteriori* filtering of maximum radius equal to 10 km.

A multivariate spatial scan statistic

First we will investigate a multivariate spatial scan statistic. In this example the temporal component of the multivariate functional data was suppressed by averaging the components over the time and we looked for spatial clusters of the combination of the different air pollutants. This will pick up areas

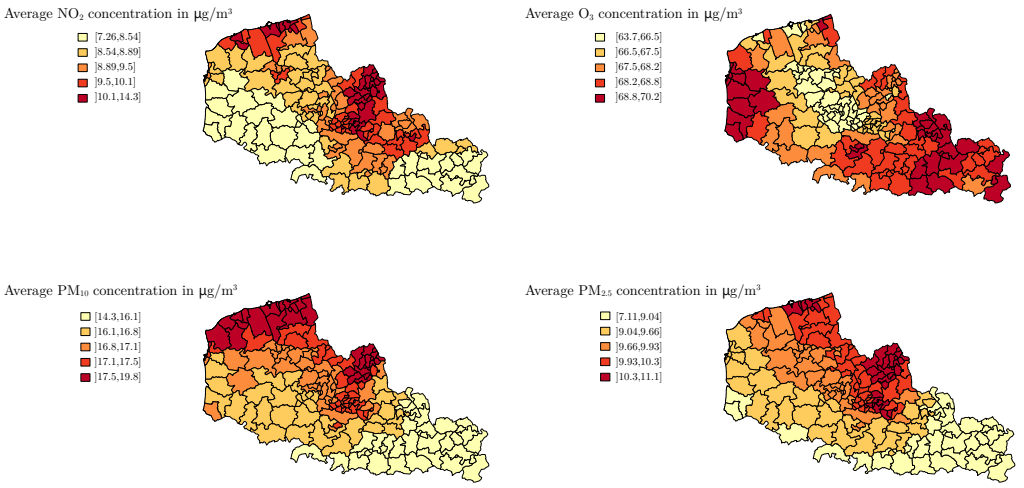


Figure 6: Spatial distributions of the average concentrations of NO_2, O_3, PM_{10} and $PM_{2.5}$ over the period from May 1, 2020 to June 25, 2020. A spatial heterogeneity of the average concentration (over the study period) is observed for the four pollutants. The spatial distributions of the average concentrations of PM_{10} and $PM_{2.5}$ are also observed to be similar.

of multiple exposure to pollutants or, on the contrary, areas with little pollution. We first checked the normality of each variable, which has been done by using a histogram and a qqplot. Since the distribution of the pollutants temporal mean concentrations is non-normal we decide to apply the MNP scan procedure. Here the system of coordinates is "WGS84", it must be filled with the argument `system`. As explained in Section Computing the spatial scan statistic, Kulldorff and Nagarwalla (1995) recommended to consider a maximum size of half the study region for the potential clusters so we use this *a priori* filtering with the parameters `mini`, `maxi` and `type_minimaxi`: the potential clusters are circular and they contain between 1 and 50% of the sites. Then as noticed in Section Air pollution in northern France, we will apply an *a posteriori* filtering of maximum radius equal to 10 km (arguments `mini_post`, `maxi_post` and `type_minimaxi_post`). Here we only want to consider the significant clusters at the 5% threshold. Thus we leave the `typeI` parameter at its default value (0.05). However it should be noted that it is possible to obtain all the clusters (the MLC and the secondary clusters (Kulldorff, 1997)) by setting the `typeI` value at 1.

```
library(sp)
coords <- coordinates(map_sites)
res_mnp <- SpatialScan(method = "MNP", data = multi_data, sites_coord = coords,
+ system = "WGS84", mini = 1, maxi = nrow(coords)/2, type_minimaxi = "sites/indiv",
+ mini_post = 0, maxi_post = 10, type_minimaxi_post = "radius",
+ nbCPU = 7, MC = 99, variable_names = c("NO2", "O3", "PM10", "PM2.5"))$MNP
```

Once the scan procedure is completed, the plot function can be used. For brevity, we only focus on the MLC and for the sake of completeness we will show the use of the three possible visualizations of the clusters. Since we have a spatial object `map_sites` we can use the types `"map"` and `"map2"`. However for sake of completeness we also show the use of `"schema"` which allows to display the clusters otherwise (Figure 7). For the latter, since the system of the coordinates is "WGS84", the plot function requires to complete the parameter `system_conv` which allows to correctly project the points. Here we choose the EPSG code 2154 corresponding to the Lambert 93 projection since the data is located in metropolitan France.

```
plot(x = res_mnp, type = "map", spobject = map_sites, only.MLC = TRUE)
plot(x = res_mnp, type = "map2", spobject = map_sites, only.MLC = TRUE)
plot(x = res_mnp, type = "schema", system_conv = "+init=epsg:2154", only.MLC = TRUE)
```

Finally users may want to get some summarized characteristics, such as the quantiles of the variables. This can be achieved by using the function `summary` with the argument `type_summ` equal to `"nparam"` (for the quantiles):

```
summary(res_mnp, type_summ = "nparam", only.MLC = TRUE)

## $basic_summary
##         Cluster 1
```

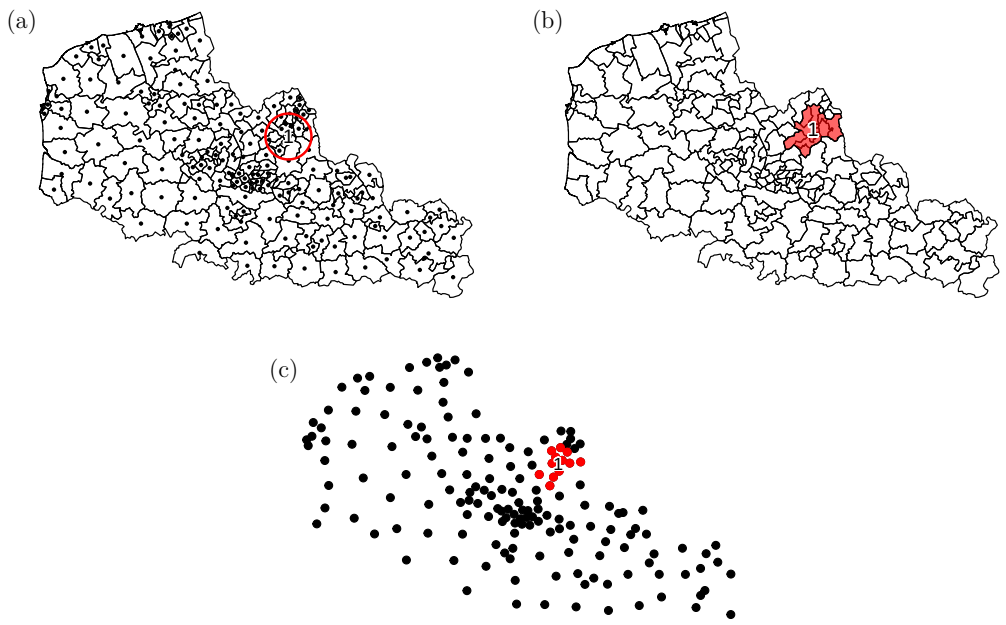


Figure 7: Visualization of the most likely cluster with the function `plot` with the types `"map"` (panel a), `"map2"` (panel b) and `"schema"` (panel c) for the MNP scan procedure computed with the function `SpatialScan` on the average concentrations of NO_2, O_3, PM_{10} and $PM_{2.5}$. The first two types of visualization require a spatial object corresponding to the spatial units (here the 169 *cantons* of the *Nord-Pas-de-Calais* region of northern France). The visualization option `"schema"` allows in the other case to draw a schema of the spatial units and the detected clusters. Here we focus the cluster visualization on the most likely cluster which is located in the urban area of Lille.

```
## p-value     0.001
## Radius      9.999
##
## $complete_summary
##                 Overall Inside cluster 1 Outside cluster 1
## Number of sites 169.000           12.000           157.000
## Q25 NO2           8.673           11.327             8.635
## Median NO2        9.183           11.721             9.075
## Q75 NO2           9.848           12.382             9.692
## Q25 O3           66.778           67.527            66.721
## Median O3        67.895           67.609            67.961
## Q75 O3           68.564           67.922            68.658
## Q25 PM10         16.397           17.483            16.205
## Median PM10      16.970           17.877            16.933
## Q75 PM10         17.372           17.962            17.266
## Q25 PM2.5         9.132           10.584             9.113
## Median PM2.5      9.833           10.678             9.790
## Q75 PM2.5        10.213           10.919            10.107
```

The user can also use the function `plotSummary` to display the spider chart corresponding to the detected cluster (Figure 8).

```
plotSummary(res_mnp, type = "median", only.MLC = TRUE)
```

The MLC is located in the area of Lille. The summary and Figure 8 show that it is a cluster of overpollution (except for the pollutant O_3). This cluster is especially characterized by high concentrations of NO_2 and $PM_{2.5}$ which indicates pollution from road traffic and from the residential sector (auxiliary heating in particular). As the adverse health effects of air pollution (and their potential synergistic effect) are well established, such a result could inform local stakeholders about immediate interventions around the area of Lille to reduce the air pollution levels.

We have obtained some first results however the curves on Figure 5 present a marked temporal

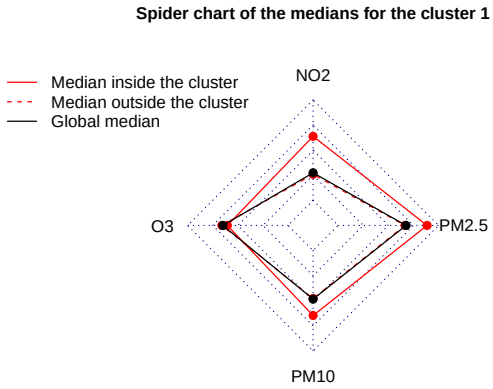


Figure 8: Spider chart obtained with the function `plotSummary` for the most likely cluster detected on the average concentrations of NO_2, O_3, PM_{10} and $PM_{2.5}$ by the MNP scan procedure in northern France. The most likely cluster is characterized by larger median concentrations of NO_2, PM_{10} and $PM_{2.5}$ than outside the cluster.

variability during the study period. Thus it could be interesting to apply functional spatial scan statistics.

A univariate functional spatial scan statistic

Here we only consider the pollutant NO_2. Applying a spatial scan statistic for univariate functional data will thus allow to highlight areas where the NO_2 concentration curves are abnormally high or, on the contrary abnormally low. We choose to use the URBFSS scan procedure since it often presents higher powers and true positive rates than the other univariate functional methods as its multivariate counterpart MRBFSS (Frévent et al., 2021b). As mentioned in Section A multivariate spatial scan statistic we decide to use the set of potential clusters *a priori* in the Equation 1 which corresponds to the recommended approach of Kulldorff and Nagarwalla (1995), and to the default values of the parameters `mini`, `maxi` and `type_minimaxi` in the scan functions. We also set a maximum radius equal to 10 km *a posteriori*.

```
res_urbfss <- SpatialScan(method = "URBFSS", data = funi_data, sites_coord = coords,
+ system = "WGS84", mini = 1, maxi = nrow(coords)/2, type_minimaxi = "sites/indiv",
+ mini_post = 0, maxi_post = 10, type_minimaxi_post = "radius",
+ nbCPU = 7, MC = 99)$URBFSS
plot(res_urbfss, type = "map2", spobject = map_sites, only.MLC = TRUE)
```

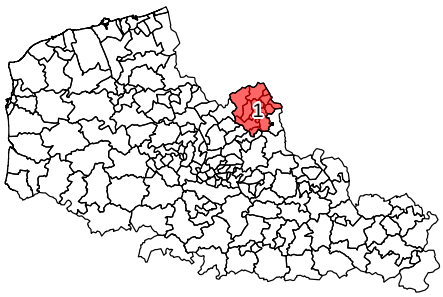


Figure 9: Visualization of the detected clusters with the function `plot` with `type = "map2"` for the URBFSS scan procedure computed using the concentrations of NO_2 in the 169 *cantons* of the *Nord-Pas-de-Calais* region of northern France over the period from May 1, 2020 to June 25, 2020. Here we focus the cluster visualization on the most likely cluster which is located in the urban area of Lille.

Again the MLC is located in the area of Lille (Figure 9).

For functional data another function is provided to give some characteristics of the clusters: we can visualize the curves in the cluster by adding the curve of the global median with the function

plotCurves. The function plotSummary allows to visualize the median curves inside and outside the cluster (Figure 10): this is a cluster of overexposure to NO_2, which indicates traffic-related air pollution. Since exposure to pollution impacts health negatively, these results can be used to intervene to reduce air pollution.

```
plotCurves(res_urbfss, add_median = TRUE, only.MLC = TRUE)
plotSummary(res_urbfss, type = "median", only.MLC = TRUE)
```

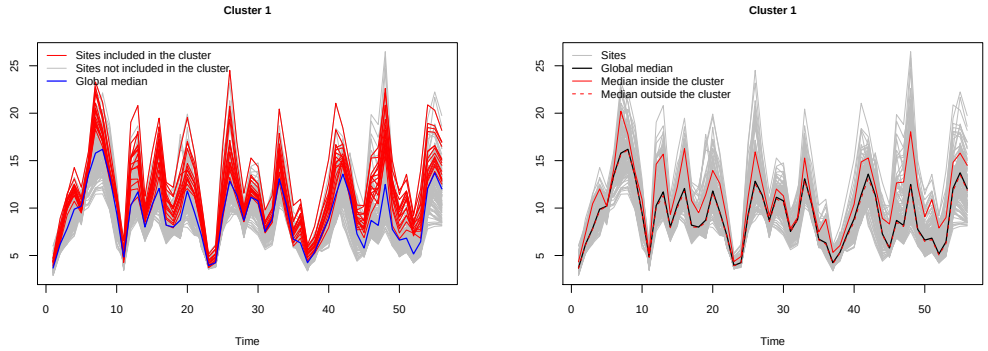


Figure 10: Characterization of the most likely cluster detected by the URBFSS scan approach in the context of univariate functional data consisting of the NO_2 concentrations in northern France over the period from May 1, 2020 to June 25, 2020 with the functions plotCurves (left panel) and plotSummary (right panel). The most likely cluster is characterized by high concentration curves with a median concentration curve higher than outside the cluster.

A functional multivariate spatial scan statistic

Now we consider the four pollutants together. To detect spatial clusters of the combination of the four pollutants considering all available information on the time period, we apply a spatial scan statistic for multivariate functional data. It will identify geographical areas in which one or more of the pollutant concentration curves are abnormally high or abnormally low. For the same reason that we have previously chosen to apply the URBFSS scan procedure, we use the MRBFSS in this context, with the same restrictions *a priori* and *a posteriori* as for the MNP and the URBFSS scan approaches.

```
res_mrbfss <- SpatialScan(method = "MRBFSS", data = fmulti_data, sites_coord = coords,
+ system = "WGS84", mini = 1, maxi = nrow(coords)/2, type_minimaxi = "sites/indiv",
+ mini_post = 0, maxi_post = 10, type_minimaxi_post = "radius",
+ nbCPU = 7, MC = 99, variable_names = c("NO2", "O3", "PM10", "PM2.5"))$MRBFSS
plot(res_mrbfss, type = "map2", spobject = map_sites, only.MLC = TRUE)
```

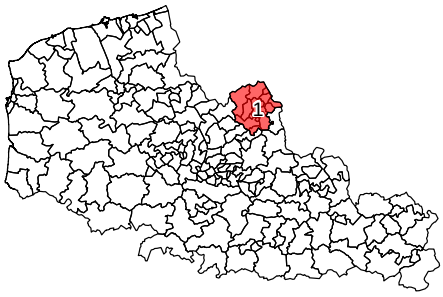


Figure 11: Visualization of the most likely cluster with the function plot with type = "map2" for the MRBFSS scan procedure computed using the concentrations of NO_2, O_3, PM_{10} and $PM_{2.5}$ in the 169 *cantons* of the *Nord-Pas-de-Calais* region of northern France over the period from May 1, 2020 to June 25, 2020. The most likely cluster is located in the urban area of Lille.

The detected cluster is exactly the same as before and is therefore located in the urban area of Lille (Figure 11).

Again we will display the curves in the cluster by adding the curve of the global median (Figure 12), as well as the median curves inside and outside the cluster which show that this is a cluster of high concentrations of NO_2, PM_{10} and $PM_{2.5}$ (Figure 13). As mentioned in Section A multivariate spatial scan statistic, in environmental science it is well-known that NO_2 and $PM_{2.5}$ are more frequent in urban areas due to road traffic and population density so this is consistent with the cluster observed here. As the adverse health effects of air pollution and the combined effects of air pollutants are well established, this result could enable interventions by local authorities around the Lille area to reduce air pollution.

```
plotCurves(res_mrbfss, add_median = TRUE, only.MLC = TRUE)
plotSummary(res_mrbfss, type = "median", only.MLC = TRUE)
```

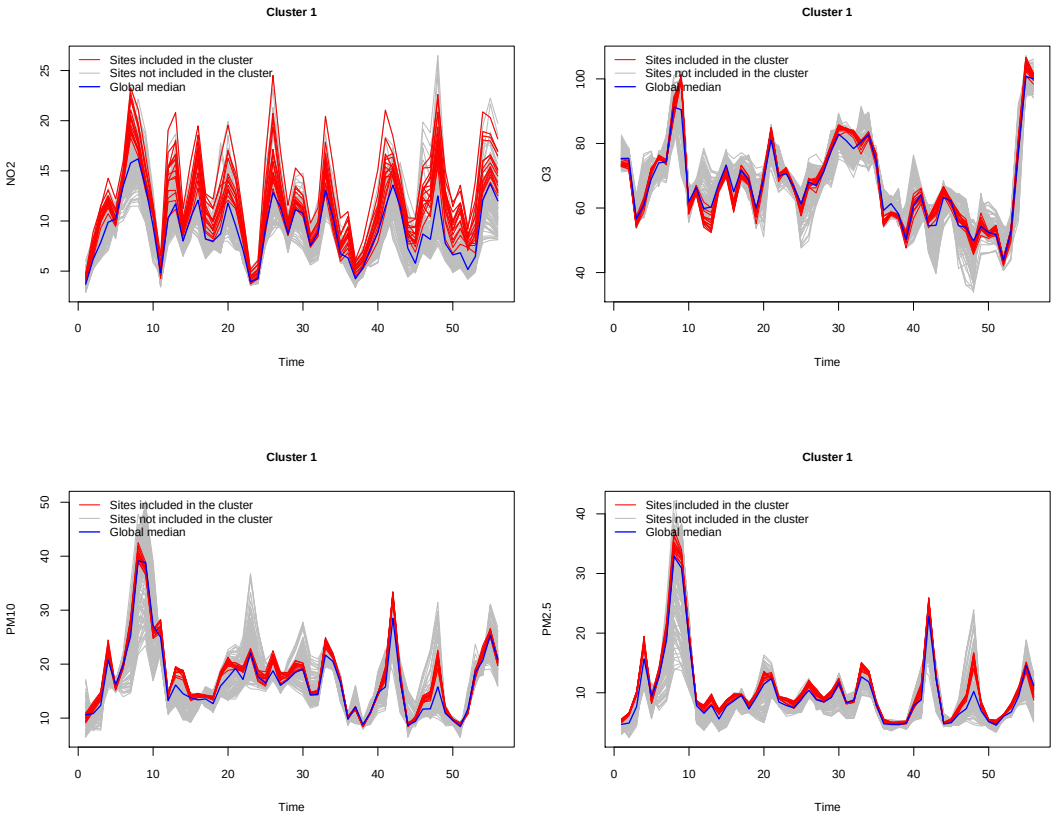


Figure 12: Characterization of the most likely cluster detected by the MRBFSS scan approach in the context of multivariate functional data consisting of the NO_2, O_3, PM_{10} and $PM_{2.5}$ concentrations in northern France over the period from May 1, 2020 to June 25, 2020 with the function `plotCurves`. The most likely cluster is characterized by high concentration curves of NO_2, PM_{10} and $PM_{2.5}$.

5 Summary

In this article we presented the **HDSpatialScan** package. It makes it very easy to apply the existing scan statistics developed for multivariate data or functional data (univariate or multivariate), and the new rank-based scan statistic for univariate functional data presented in the Section Models. The potential clusters considered are of variable size and circular. In further updates of the package **HDSpatialScan** other shapes of scanning window such as elliptical or rectangular shapes will be implemented. Our package also allows to easily plot and summarize the detected clusters. Then examples of applications of the functions of the package have been shown. **HDSpatialScan** presents the advantage that all the scan procedures are applied using the same function `SpatialScan` and it uses the classical R functions `plot`, `print` and `summary` which makes it very quick to get started.

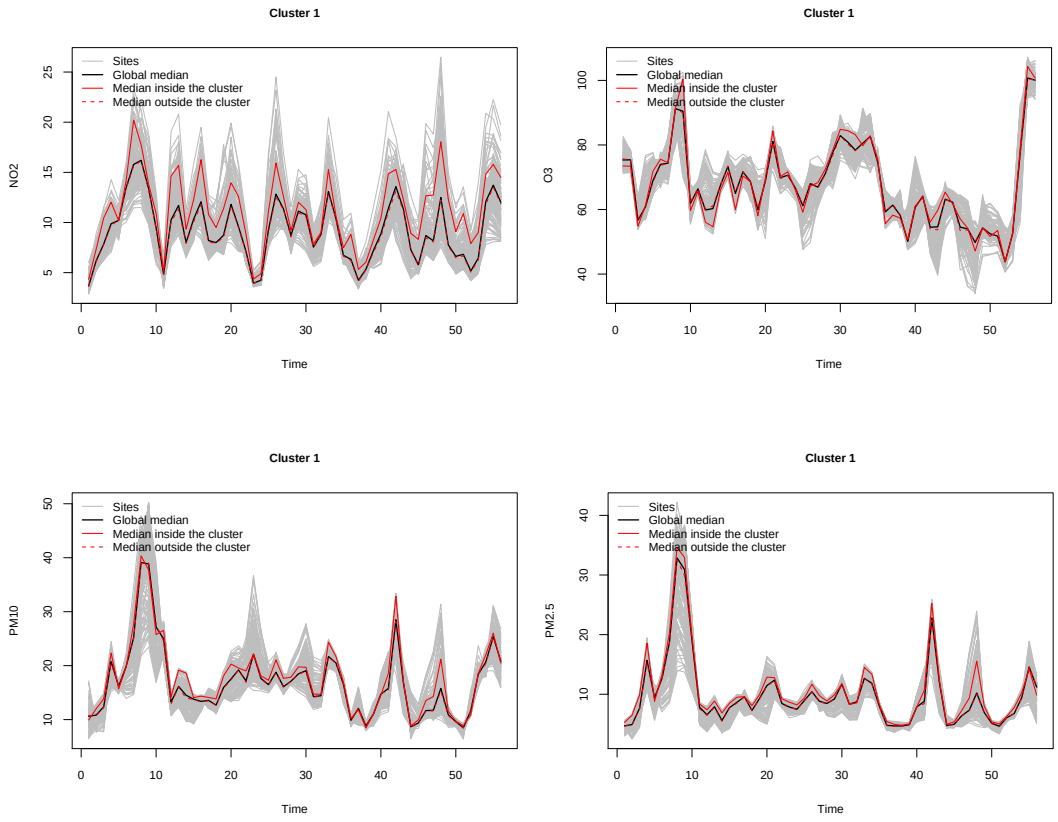


Figure 13: Characterization of the most likely cluster detected by the MRBFSS scan approach in the context of multivariate functional data consisting of the NO_2, O_3, PM_{10} and $PM_{2.5}$ concentrations in northern France over the period from May 1, 2020 to June 25, 2020 with the function `plotSummary`. The most likely cluster is characterized by median concentration curves of NO_2, PM_{10} and $PM_{2.5}$ higher than outside the cluster.

Bibliography

B. Allévius. scanstatistics: Space-Time anomaly detection using scan statistics. *Journal of Open Source Software*, 3, 2018a. URL https://doi.org/10.21105/joss.00515. [p96]

B. Allévius. *scanstatistics: Space-Time Anomaly Detection using Scan Statistics*, 2018b. URL https://CRAN.R-project.org/package=scanstatistics. [p96]

J. Berrendero, A. Justel, and M. Svarc. Principal components for multivariate functional data. *Computational Statistics & Data Analysis*, 55:2619–2634, 2011. URL https://doi.org/10.1016/j.csda.2011.03.011. [p96]

V. Bhatt and N. Tiwari. A spatial scan statistic for survival data based on Weibull distribution. *Statistics in medicine*, 33(11):1867–1876, 2014. URL https://doi.org/10.1002/sim.6075. [p95]

G. Boente and R. Fraiman. Kernel-based functional principal components. *Statistics & Probability Letters*, 48:335–345, 2000. URL https://doi.org/10.1016/S0167-7152(00)00014-6. [p96]

A. Chakraborty and P. Chaudhuri. A Wilcoxon-Mann-Whitney type test for infinite dimensional data. *Biometrika*, 102, 2014. URL https://doi.org/10.1093/biomet/asu072. [p100, 101]

C. Chen, A. Y. Kim, M. Ross, and J. Wakefield. *SpatialEpi: Methods and Data for Spatial Epidemiology*, 2018. URL https://CRAN.R-project.org/package=SpatialEpi. [p96]

J.-M. Chiou and H.-G. Müller. Diagnostics for functional regression via residual processes. *Computational Statistics & Data Analysis*, 15:4849–4863, 06 2007. URL https://doi.org/10.1016/j.csda.2006.07.042. [p96]

L. Cucala. A distribution-free spatial scan statistic for marked point processes. *Spatial Statistics*, 10:117–125, 2014. URL https://doi.org/10.1016/j.spasta.2014.03.004. [p99, 101]

L. Cucala. A Mann-Whitney scan statistic for continuous data. *Communications in Statistics-Theory and Methods*, 45(2):321–329, 2016. URL https://doi.org/10.1080/03610926.2013.806667. [p95]

L. Cucala, C. Demattei, P. Lopes, and A. Ribeiro. A spatial scan statistic for case event data based on connected components. *Computational Statistics*, 28(1):357–369, 2013. URL https://doi.org/10.1007/s00180-012-0304-6. [p97]

L. Cucala, M. Genin, C. Lanier, and F. Occelli. A multivariate Gaussian scan statistic for spatial data. *Spatial Statistics*, 21:66–74, 2017. URL https://doi.org/10.1016/j.spasta.2017.06.001. [p96, 98, 105]

L. Cucala, M. Genin, F. Occelli, and J. Soula. A multivariate nonparametric scan statistic for spatial data. *Spatial statistics*, 29:1–14, 2019. URL https://doi.org/10.1016/j.spasta.2018.10.002. [p96, 99, 105]

A. Cuevas, M. Febrero-Bande, and R. Fraiman. Linear functional regression: The case of fixed design and functional response. *The Canadian Journal of Statistics*, 30:285–300, 2002. URL https://doi.org/10.2307/3315952. [p96]

A. Cuevas, M. Febrero-Bande, and R. Fraiman. An ANOVA test for functional data. *Computational Statistics & Data Analysis*, 47:111–122, 2004. URL https://doi.org/10.1016/j.csda.2003.10.021. [p100]

C. Dematteï, N. Molinari, and J.-P. Daurès. SPATCLUS: An R package for arbitrarily shaped multiple spatial cluster detection for case event data. *Computer Methods and Programs in Biomedicine*, 84(1):42–49, 2006. ISSN 0169-2607. URL https://doi.org/10.1016/j.cmpb.2006.07.008. [p96]

H. Durbeck, D. Greiling, L. Estberg, A. Long, G. Jacquez, Y. Pallicaris, and S. Hinton. *ClusterSeer: Software for the Detection and Analysis of Event Clusters, User Manual Book 2, Version 2.5*, 2012. [p96]

M. Dwass. Modified randomization tests for nonparametric hypotheses. *The Annals of Mathematical Statistics*, 28(1):181–187, 1957. URL https://doi.org/10.1214/aoms/1177707045. [p105]

F. Ferraty and P. Vieu. The functional nonparametric model and application to spectrometric data. *Computational Statistics*, 17:545–564, 2002. URL https://doi.org/10.1007/s001800200126. [p96]

C. Frévent, M.-S. Ahmed, M. Marbac, and M. Genin. Detecting spatial clusters in functional data: New scan statistic approaches. *Spatial Statistics*, page 100550, 2021a. URL https://doi.org/10.1016/j.spasta.2021.100550. [p96, 100, 101, 104, 105]

C. Frévent, M.-S. Ahmed, S. Dabo-Niang, and M. Genin. Investigating spatial scan statistics for multivariate functional data. arXiv:2103.14401v1, 2021b. [p96, 101, 103, 104, 105, 113]

J. Gao, Z. Zhang, Y. Hu, J. Bian, W. Jiang, X. Wang, L. Sun, and Q. Jiang. Geographical distribution patterns of iodine in drinking-water and its associations with geological factors in Shandong province, China. *International Journal of Environmental Research and Public Health*, 11(5):5431–5444, 2014. URL https://doi.org/10.3390/ijerph110505431. [p95]

M. Genin, M. Fumery, F. Occelli, G. Savoye, B. Pariente, L. Dauchet, J. Giovannelli, C. Vignal, M. Body-Malapel, H. Sarter, et al. Fine-scale geographical distribution and ecological risk factors for Crohn's disease in France (2007-2014). *Alimentary Pharmacology & Therapeutics*, 51(1):139–148, 2020. URL https://doi.org/10.1111/apt.15512. [p95]

V. Gómez-Rubio, J. Ferrándiz-Ferragud, and A. López-Quílez. *DCluster: Functions for the Detection of Spatial Clusters of Diseases*, 2015. URL https://CRAN.R-project.org/package=DCluster. [p96]

V. Gómez-Rubio, P. Moraga, J. Molitor, and B. Rowlingson. DClusterm: Model-based detection of disease clusters. *Journal of Statistical Software*, 90(14):1–26, 2019. URL https://doi.org/10.18637/jss.v090.i14. [p96]

V. Gomez-Rubio, P. E. M. Serrano, and B. Rowlingson. *DClusterm: Model-Based Detection of Disease Clusters*, 2020. URL https://CRAN.R-project.org/package=DClusterm. [p96]

T. Górecki and Ł. Smaga. A comparison of tests for the one-way ANOVA problem for functional data. *Computational Statistics*, 30:987–1010, 2015. URL https://doi.org/10.1007/s00180-015-0555-0. [p100]

T. Górecki and Ł. Smaga. Multivariate analysis of variance for functional data. *Journal of Applied Statistics*, 44(12):2172–2189, 2017. URL https://doi.org/10.1080/02664763.2016.1247791. [p103]

D. Greiling, L. Estberg, A. Long, and G. Jacquez. *ClusterSeer: Software for the Detection and Analysis of Event Clusters, User Manual Book 1, Version 2.5*, 2012. [p96]

L. Huang, M. Kulldorff, and D. Gregorio. A spatial scan statistic for survival data. *Biometrics*, 63(1): 109–118, 2007. URL https://doi.org/10.1111/j.1541-0420.2006.00661.x. [p95]

I. Jung and H. J. Cho. A nonparametric spatial scan statistic for continuous data. *International Journal of Health Geographics*, 14, 2015. URL https://doi.org/10.1186/s12942-015-0024-6. [p95, 99, 101, 102, 105]

I. Jung, M. Kulldorff, and A. C. Klassen. A spatial scan statistic for ordinal data. *Statistics in Medicine*, 26(7):1594–1607, 2007. URL https://doi.org/10.1002/sim.2607. [p95]

M. M. Khan, S. Roberson, K. Reid, M. Jordan, and A. Odoi. Geographic disparities and temporal changes of diabetes prevalence and diabetes self-management education program participation in Florida. *Plos one*, 16(7), 2021. URL https://doi.org/10.1371/journal.pone.0254579. [p95]

K. Kleinman. *rsatscan: Tools, Classes, and Methods for Interfacing with SaTScan Stand-Alone Software*, 2015. URL https://CRAN.R-project.org/package=rsatscan. [p96]

M. Kulldorff. A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26:1481–1496, 1997. URL https://doi.org/10.1080/03610929708831995. [p95, 109, 111]

M. Kulldorff. *TreeScan User Guide*, 2018. URL https://www.treescan.org/. [p96]

M. Kulldorff. *SaTScan User Guide for Version 9.7*, 2021. URL https://www.satscan.org/. [p96]

M. Kulldorff and N. Nagarwalla. Spatial disease clusters: Detection and inference. *Statistics in Medicine*, 14(8):799–810, 1995. URL https://doi.org/10.1002/sim.4780140809. [p95, 97, 108, 111, 113]

M. Kulldorff, W. Athas, E. Feurer, B. Miller, and C. Key. Evaluating cluster alarms: A space-time scan statistic and brain cancer in Los Alamos, New Mexico. *The American Journal of Public Health*, 88(9): 1377–1380, 1998. URL https://doi.org/10.2105/AJPH.88.9.1377. [p96]

M. Kulldorff, Z. Fang, and S. J. Walsh. A tree-based scan statistic for database disease surveillance. *Biometrics*, 59(2):323–331, 2003. URL https://doi.org/10.1111/1541-0420.00039. [p96]

M. Kulldorff, L. Huang, L. Pickle, and L. Duczmal. An elliptic spatial scan statistic. *Statistics in Medicine*, 25:3929–3943, 2006. URL https://doi.org/10.1002/sim.2490. [p97]

M. Kulldorff, F. Mostashari, L. Duczmal, W. K. Yih, K. Kleinman, and R. Platt. Multivariate scan statistics for disease surveillance. *Statistics in Medicine*, 26:1824–1833, 2007. URL https://doi.org/10.1002/sim.2818. [p95, 96]

M. Kulldorff, L. Huang, and K. Konty. A scan statistic for continuous data based on the normal probability model. *International Journal of Health Geographics*, 8(58), 2009. URL https://doi.org/10.1186/1476-072X-8-58. [p95, 99, 105]

Z. Lin, M. E. Lopes, and H.-G. Müller. High-dimensional MANOVA via bootstrapping and its application to functional and sparse count data. *Journal of the American Statistical Association*, 2021. URL https://doi.org/10.1080/01621459.2021.1920959. [p101]

R. Loche, B. Giron, D. Abrial, L. Cucala, M. CharrasGarrido, and J. De-Goer. *graphscan: Cluster Detection with Hypothesis Free Scan Statistic*, 2016. URL https://CRAN.R-project.org/package=graphscan. [p96]

L. H. S. C. Marciano, A. de Faria Fernandes Belone, P. S. Rosa, N. M. B. Coelho, C. C. Ghidella, S. M. T. Nardi, W. C. Miranda, L. V. Barrozo, and J. C. Lastória. Epidemiological and geographical characterization of leprosy in a Brazilian hyperendemic municipality. *Cadernos de saude publica*, 34, 2018. URL https://doi.org/10.1590/0102-311X00197216. [p95]

P. Moraga. *SpatialEpiApp: A Shiny Web Application for the Analysis of Spatial and Spatio-Temporal Disease Data*, 2017a. URL https://CRAN.R-project.org/package=SpatialEpiApp. [p96]

P. Moraga. SpatialEpiApp: A Shiny Web Application for the Analysis of Spatial and Spatio-Temporal Disease Data. *Spatial and Spatio-temporal Epidemiology*, 23, 2017b. URL https://doi.org/10.1016/j.sste.2017.08.001. [p96]

H. Oja and R. H. Randles. Multivariate nonparametric tests. *Statistical Science*, 18(4):598–605, 11 2004. URL https://doi.org/10.1214/088342304000000558. [p99, 104, 108]

T. Otani and K. Takahashi. *rflexscan: The Flexible Spatial Scan Statistic*, 2021. URL https://CRAN.R-project.org/package=rflexscan. [p96]

PROJ contributors. *PROJ coordinate transformation software library*. Open Source Geospatial Foundation, 2021. URL https://proj.org/. [p109]

Z. Qiu, J. Chen, and J.-T. Zhang. Two-sample tests for multivariate functional data with applications. *Computational Statistics & Data Analysis*, 157, 2021. URL https://doi.org/10.1016/j.csda.2020.107160. [p104]

J. Ramsay and B. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer-Verlag, 2005. [p96, 108]

J. O. Ramsay, S. Graves, and G. Hooker. *fda: Functional Data Analysis*, 2020. URL https://CRAN.R-project.org/package=fda. [p108]

G. Shi, J. Liu, and X. Zhong. Spatial and temporal variations of $PM_{2.5}$ concentrations in Chinese cities during 2015-2019. *International Journal of Environmental Health Research*, pages 1–13, 2021. URL https://doi.org/10.1080/09603123.2021.1987394. [p95]

Z. Smida, L. Cucala, A. Gannoun, and G. Durif. A Wilcoxon-Mann-Whitney spatial scan statistic for functional data. *Computational Statistics & Data Analysis*, 167:107378, 2022. URL https://doi.org/10.1016/j.csda.2021.107378. [p96, 100, 101, 105]

D. L. Sudakin, Z. Horowitz, and S. Giffin. Regional variation in the incidence of symptomatic pesticide exposures: Applications of geographic information systems. *Journal of Toxicology: Clinical Toxicology*, 40(6):767–773, 2002. URL https://doi.org/10.1081/CLT-120015837. [p95]

K. Takahashi and T. Tango. A flexibly shaped spatial scan statistic for detecting clusters. *International Journal of Health Geographics*, 4(11), 05 2005. URL https://doi.org/10.1186/1476-072X-4-11. [p96]

K. Takahashi, T. Yokoyama, and T. Tango. *FleXScan v 3.1: Software for the Flexible Scan Statistic*, 2010. [p96]

L. Wan, Y. Sun, I. Lee, W. Zhao, and F. Xia. Industrial pollution areas detection and location via satellite-based IIoT. *IEEE Transactions on Industrial Informatics*, 17(3):1785–1794, 2020. URL https://doi.org/10.1109/TII.2020.2992658. [p95]

Camille Frévent
University of Lille, CHU Lille, ULR 2694 - METRICS: Évaluation des technologies de santé et des pratiques médicales, INRIA-MODAL
F-59000 Lille France
camille.frevent@univ-lille.fr

Mohamed-Salem Ahmed
University of Lille, CHU Lille, ULR 2694 - METRICS: Évaluation des technologies de santé et des pratiques médicales
F-59000 Lille France

Julien Soula
University of Lille, CHU Lille, ULR 2694 - METRICS: Évaluation des technologies de santé et des pratiques médicales
F-59000 Lille France

Lionel Cucala
IMAG, Université de Montpellier, CNRS
Montpellier
France

Zaineb Smida
IMAG, University of Montpellier, CNRS
Montpellier
France

Sophie Dabo-Niang
Laboratoire Paul Painvelé UMR CNRS 8524, INRIA-MODAL, University of Lille
F-59000 Lille France

Michaël Genin
University of Lille, CHU Lille, ULR 2694 - METRICS: Évaluation des technologies de santé et des pratiques médicales
F-59000 Lille France

HDiR: An R Package for Computation and Nonparametric Plug-in Estimation of Directional Highest Density Regions and General Level Sets

by Paula Saavedra-Nieves, Rosa M. Crujeiras

Abstract A deeper understanding of a distribution support, being able to determine regions of a certain (possibly high) probability content is an important task in several research fields. Package **HDiR** for R is designed for exact computation of directional (circular and spherical) highest density regions and density level sets when the density is fully known. Otherwise, **HDiR** implements nonparametric plug-in methods based on different kernel density estimates for reconstructing this kind of sets. Additionally, it also allows the computation and plug-in estimation of level sets for general functions (not necessarily a density). Some exploratory tools, such as suitably adapted distances and scatterplots, are also implemented. Two original datasets and spherical density models are used for illustrating **HDiR** functionalities.

1 An overview on directional general level sets and highest density regions

When analyzing a data distribution, it is often the case that for a deeper understanding of the modelling problem, it is interesting to determine regions on the density support exceeding a certain threshold on the density function values. These regions are known as density level sets and, if the density is unknown, such a task can be accomplished from a set estimation perspective. Set estimation deals with the problem of reconstructing a set (or estimating any of its features such as its boundary or its volume) from a random sample of points intimately related to it. Since Hartigan (1975) establishes the notion of clusters as connected components of a density level set, the reconstruction of this particular type of sets has been widely considered in the literature (mainly for densities supported on an Euclidean space). There are only very few contributions where density level set theory has been extended to more general domains such as the unit sphere or manifolds. Cuevas et al. (2006) consider the estimation of level sets for general functions (not necessarily a density) such as regression curves, providing some consistency theoretical results and showing a density level set on the sphere for illustration. More recently, the reconstruction of density level sets on manifolds is studied in Cholaquidis et al. (2022), who also presents some simulations illustrating the performance of their approach on the torus and on the sphere.

Let X be a random vector taking values on a $d-$dimensional unit sphere S^{d-1} with density f and $t > 0$, the goal of (directional) density level set estimation is to reconstruct the set

$$G_f(t) = \{x \in S^{d-1} : f(x) \geq t\}. \tag{1}$$

from a random sample of points $\mathcal{X}_n = \{X_1, \cdots, X_n\}$ of X when f is unknown. As an illustration, some (theoretical) level sets are shown in Figure 1 by representing $G_f(t)$ for a circular (left) and a spherical density (right) when three different values of the level t are chosen. The threshold t is represented through a dotted line for the circular case. Note that, if large values of t are considered, $G_f(t)$ coincides with the greatest modes of the circular/spherical distribution. However, for small values of t, the level set $G_f(t)$ is practically equal to the support of the distribution. Therefore, cluster definition via connected components in Hartigan (1975) is clearly related to the notion of mode. Note also that the computation of the number of modes considering the values of a density over a certain range of values for the level t, would enable the construction of a directional cluster tree. Azzalini and Torelli (2007) already present this statistical tool for Euclidean data. Moreover, the association between clusters and modes is the basis of modal clustering methodology (see Menardi, 2016 for a review on this topic). Most modal clustering algorithms are based on the application of a mode-seeking numerical method to the sample points and assigning the same cluster to those data that are iteratively shifted to the same limit value. Examples of such procedures include the mean shift algorithm that has been already studied in S^{d-1} (see, for instance, Chang-Chien et al., 2010 and Yang et al., 2014).

Despite a practitioner may be interested in determining this type of regions, the value of the level t could be (in principle) unknown in real situations. In practice, it is quite common to assume that the set in (1) must satisfy a probability content previously established. Following Box and Tiao

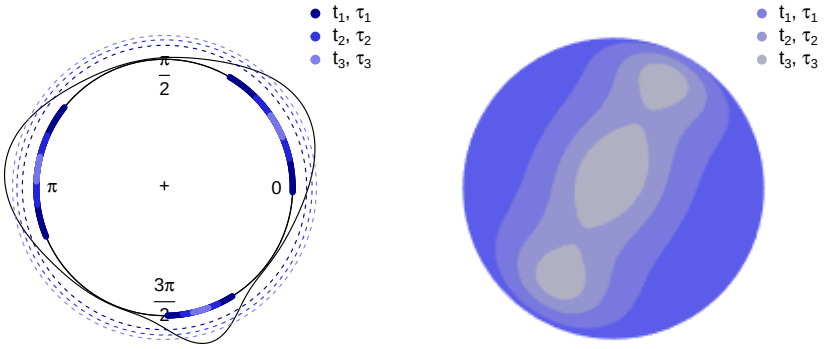


Figure 1: For a circular density (left) and a spherical density (right), level set $G_f(t)$ for $t = t_1$, $t = t_2$ and $t = t_3$ verifying $0 < t_1 < t_2 < t_3$. Equivalently, HDR $L(f_\tau)$ for $\tau = \tau_1 = 0.2$, $\tau = \tau_2 = 0.5$ and $\tau = \tau_3 = 0.8$.

(1973), Hyndman (1996) and, more recently, Azzalini and Torelli (2007), Saavedra-Nieves and Crujeiras (2021b) generalize the definition of HDRs from the Euclidean to the directional setting, providing a plug-in estimation method. Specifically, HDRs are a kind of density level sets where the set probability content is fixed instead of the level t. The estimation of HDRs involves further complexities given that the threshold must be computed from the previously fixed probability content. Formally, given $\tau \in (0, 1)$, the $100(1-\tau)\%$ HDR is the subset

$$L(f_\tau) = \{x \in S^{d-1} : f(x) \geq f_\tau\}, \tag{2}$$

where f_τ can be seen as the largest constant such that

$$\mathbb{P}(X \in L(f_\tau)) \geq 1 - \tau,$$

with respect to the distribution induced by f. Figure 1 also shows the HDR $L(f_\tau)$ for a circular and a spherical densities with three different values of τ. Note that, if large values of τ are considered, $L(f_\tau)$ is equal to the greatest modes and the most distinct clusters can be easily identified. However, for small values of τ, $L(f_\tau)$ is almost equal to the support of the distribution.

To sum up, given a value of t, the computation of the level set established in (1) (and of its connected components) is a quite simple mathematical task when f is known. Under this assumption and taking a fixed $\tau \in (0, 1)$, determining the HDR introduced in (2) presents a similar complexity but, in this case, it is additionally necessary to determine the threshold f_τ. In particular, numerical integration methods can be applied to solve that problem. However, when the density f is assumed to be unknown and a random sample $\mathcal{X}_n \in S^{d-1}$ generated from f is the only available information to reconstruct the set, nonparametric set estimation techniques such as plug-in methods must be considered in order to reconstruct the connected components of the set. Perhaps due to its practical importance, Euclidean HDRs plug-in algorithms based on kernel smoothing have been widely studied even solving the problem of selecting an appropriate smoothing parameter specifically devised for the HDR reconstruction (see Baíllo and Cuevas, 2006, Samworth and Wand, 2010 or Casa et al., 2020). In the directional setting, given that a proper definition of the HDR $L(f_\tau)$ was not available, no work on this area had been carried out until the recent contribution by Saavedra-Nieves and Crujeiras (2021b).

The contents of this paper, describing the contributions in **HDiR**, mainly focus on computation and plug-in estimation of highest density regions (HDRs) and density level sets in the circle and the sphere. Although general level sets can be also analysed using **HDiR**, we will not formally detail aspects on their computation and on their plug-in reconstruction given that they can be seen as a direct generalisation of those introduced for density level sets by replacing the density by the general function under study. Therefore, with the objective of showing the capabilities of the **HDiR** package for exact computation of directional HDRs and density level sets when f is known and for plug-in estimation otherwise, this paper is organized as follows. First, a basic overview on nonparametric plug-in estimation methods is given. Initially, the classical directional kernel density estimator is briefly introduced, as it is the key tool for plug-in reconstruction and exploratory methods. Then, the problems of threshold estimation (with known and unknown density) and specific bandwidth selection for HDRs are considered. Circular confidence regions for HDRs are also established. Next, the reader will find a guided tour across **HDiR** package, illustrating its use with simulated examples first and with two real data examples later. Following the perspective in Cuevas et al. (2006), **HDiR** also allows the computation and plug-in estimation of general level sets. A reconstruction example of

a (circular) regression level set is detailed. Moreover, distances between sets and circular/spherical scatterplots are also described as exploratory tools. Finally, some discussion is provided, considering on the possible extensions of the package.

2 Plug-in estimation methods

This section provides a brief background on the design of plug-in tools included in **HDiR** for directional (circular and spherical) HDR and density level set estimation. Following Cuevas et al. (2006), if a nonparametric estimator is available for a general function, this methodology may be directly extended for reconstructing the corresponding level sets.

Plug-in estimation methods for HDRs and level sets

Although there are other nonparametric alternative routes for level set estimation, the plug-in approach has received considerable attention in the Euclidean literature (see Tsybakov, 1997, Baíllo, 2003, Mason and Polonik, 2009, Rigollet et al., 2009, Mammen and Polonik, 2013 or Chen et al., 2017). This is with no doubt a natural methodology, which can be generalized to the directional setting as in Saavedra-Nieves and Crujeiras (2021b). Given that level set estimation is a simpler problem than HDR reconstruction, we will restrict to this last setting in what follows. Given a random sample $\mathcal{X}_n \in S^{d-1}$ of the unknown directional density f, plug-in methods reconstruct the $100(1-\tau)\%$ HDR namely $L(f_\tau)$ in (2) as

$$\hat{L}(\hat{f}_\tau) = \{x \in S^{d-1} : f_n(x) \geq \hat{f}_\tau\}, \tag{3}$$

where $\hat{f}_\tau$ is an estimator of the threshold f_τ and f_n denotes a nonparametric directional density estimator. Package **HDiR** implements the kernel density estimator provided in Bai et al. (1989) ($d > 2$). From $\mathcal{X}_n$, it is defined at a point $x \in S^{d-1}$ as

$$f_n(x) = \frac{1}{n}\sum_{i=1}^{n} K_{vM}(x; X_i; 1/h^2), \tag{4}$$

where K_{vM} denotes the von Mises-Fisher kernel density and $1/h^2 > 0$ is the concentration parameter.

Following Bai et al. (1989), package **HDiR** also enables to use any kernel function (not necessarily the von Mises-Fisher density implemented by default). An example where an uniform kernel is considered will be presented later. Even more generally, **HDiR** would allow the user to define different density estimators that the one introduced in (4). See, for instance, Pelletier (2005).

As for the concentration parameter $1/h^2$, it plays an analogous role to the bandwidth in the Euclidean case. For small values of $1/h^2$, the density estimator is oversmoothed. The opposite effect is obtained as $1/h^2$ increases. Hence, the choice of h is a crucial issue. For simplicity, in what follows, we refer to h as bandwidth parameter. Many approaches for selecting h in practice, in circular and even directional settings, have been proposed in the literature (see Taylor, 2008, Oliveira et al., 2012, Hall et al., 1987, Di Marzio et al., 2011 or García-Portugués, 2013). All these existing proposals designed for density estimation are implemented in the package **NPCirc** and their aim is to minimize some error criterion on the target density. However, such a bandwidth selector may not be adequate for HDRs or level set estimation. As far as we know, such a tool was not available in the directional setting until the selector by Saavedra-Nieves and Crujeiras (2021b). It is also available in package **HDiR**. Different plug-in estimators for HDRs emerge from the consideration of all these bandwith selectors.

For the circular and the spherical densities shown in Figure 1, now Figure 2 contains the HDR plug-in estimators (bluish colours) for $\tau = 0.5$ computed using cross-validation bandwidths and samples of sizes $n = 100$ and $n = 500$, respectively. Although the theoretical circular HDR is composed by three connected components (see Figure 1), the plug-in estimator is able to detect only the two biggest clusters when $n = 100$. In order to assess the agreement of a given estimate with the theoretical target, distances between sets are the usual tools to measure the discrepancies between the theoretical sets and the corresponding empirical reconstructions. One of the most common distances in the Euclidean setting is the Hausdorff distance between the boundaries of both sets.

If the target is the reconstruction of a HDR or a density level set, the Hausdorff metric is a suitable error criterion in the directional setting (see Cuevas et al., 2006 and Cholaquidis et al., 2022). If A and B are non-empty compact sets in the $d-$dimensional Euclidean space, the Hausdorff distance between A and B is defined as follows

$$d_H(A,B) = \max\left\{\sup_{x\in A} d_E\left(\{x\},B\right), \sup_{y\in B} d_E\left(\{y\},A\right)\right\},$$

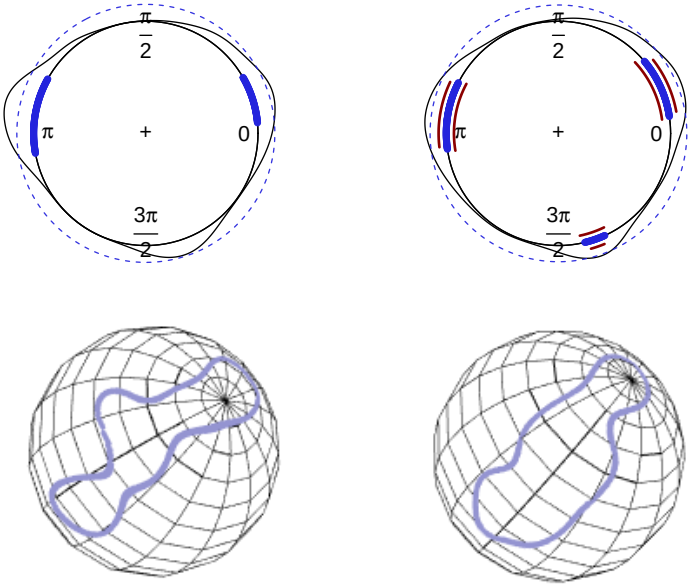


Figure 2: For the circular density shown in Figure 1 (first row), $\hat{L}(\hat{f}_\tau)$ (bluish colour) for $\tau = \tau_2 = 0.5$ computed from $\mathcal{X}_{100}$ (first column) and $\mathcal{X}_{500}$ (second column). Additionally, confidence regions are represented (dark red colour) for the second estimation. For the spherical density shown in Figure 1 (second row), $\hat{L}(\hat{f}_\tau)$ (bluish colour) for $\tau = \tau_2 = 0.5$ computed from $\mathcal{X}_{100}$ (first column) and $\mathcal{X}_{500}$ (second column).

where $d_E(\{x\}, B) = \inf_{y \in B}\{d_E(x, y)\}$ being $d_E(x, y)$ the Euclidean distance between two points. However, this metric d_H is not completely successful in detecting shape-related differences. For instance, two sets can be very close in Hausdorff distance and still show quite different shapes. This typically happens where the boundaries ∂A and ∂B are far apart, no matter the proximity of A and B. So, a natural way to reinforce the notion of visual proximity between two sets provided by Hausdorff distance is to account also for the proximity of their respective boundaries. In particular, Hausdorff distance between the boundaries of the theoretical HDR and its plug-in reconstruction is a measure of the estimation error. **HDiR** allows to compute Euclidean and Hausdorff distances between the frontiers of two arbitrary sets on the circle and on the sphere.

Threshold estimation and confidence regions for HDRs

For a given $\tau \in (0, 1)$, determining the set $L(f_\tau)$ in (2) and its plug-in estimator $\hat{L}(\hat{f}_\tau)$ in (3) involve the exact computation and the estimation of the threshold f_τ, respectively. As in the Euclidean setting, both tasks require the use of numerical integration methods. Specifically, **HDiR** uses the classical trapezoidal rule in the circular setting. However, for the spherical case, the computational cost becomes a major issue due to the complexity of the numerical integration algorithms considered on high dimensional spaces. It should be noted that package **SphericalCubature** includes some functions for solving numerical integration over spheres. However, it does not provide sufficiently accurate solutions for our problem.

An alternative approach is implemented in the internal function `sphere.integration` of **HDiR**. Specifically, the proposed numerical integration procedure on the sphere requires the definition of a triangular mesh, such as the ones depicted in Figure 3, obtained from the projection over the sphere of triangular meshes on an embedded icosaedrum. This type of mesh guarantees that there is not a prevailing direction. For computing the corresponding spherical integral, the Cartesian coordinates of the mesh vertices are transformed into spherical coordinates and standard quadrature formulae are applied in each triangle over the plane formed by the azimuthal and polar angles (see Strang and Fix, 1973).

Package **HDiR** additionally includes a computationally feasible approach for estimating f_τ in the circular and spherical context. As before, let X be a random vector with directional density f and let $Y = f(X)$ be the random vector obtained by transforming X by its own density function. Since $\mathbb{P}(f(X) \geq f_\tau) = 1 - \tau$, f_τ is exactly the $\tau-$ quantile of Y. Saavedra-Nieves and Crujeiras (2021b) establish that f_τ can be estimated as a sample quantile from a set of independent and identically distributed random vectors with the same distribution as Y. In particular, if $\mathcal{X}_n = \{X_1, \cdots, X_n\}$

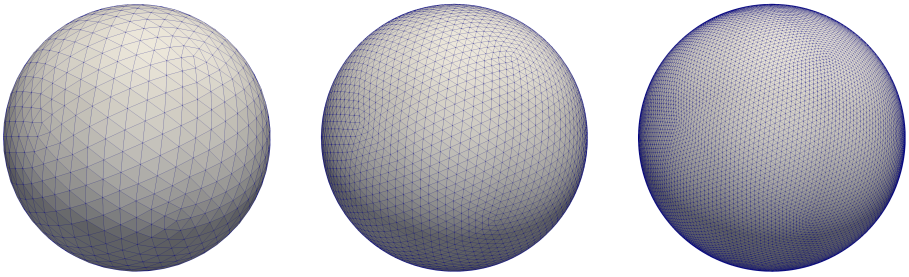


Figure 3: 3D Cartesian meshes for numerical integration on the unit sphere S^2 composed by a total of 2000 (left), 8000 (center) and 32000 (right) triangular cells.

denotes a set of independent observations in S^{d-1} from a density f, $\{f(X_1), \cdots, f(X_n)\}$ is a set of independent observations from the distribution of Y. Let $f_{(j)}$ be the $j-$th largest value of $\{f(X_i)\}_{i=1}^n$ so that $f_{(j)}$ is the (j/n) sample quantile of Y. We shall use $f_{(j)}$ as an estimate of f_τ. Specifically, we choose $\hat{f}_\tau = f_{(j)}$ where $j = \lfloor \tau n \rfloor$. Threshold values in Figure 2 were estimated following this approach.

This estimation method presents a lower computational complexity than numerical integration algorithms. Furthermore, it involves a statistical approximation. Therefore, it is possible to establish confidence intervals in order to quantify uncertainty in estimates of f_τ and, as direct consequence, to establish confidence regions for HDR's. Following Hyndman (1996), the simplest case where X is a circular random variable is considered by Saavedra-Nieves and Crujeiras (2021b). Standard asymptotic results for a sample in Cox and Hinkley (1979) ensure that $\hat{f}_\tau$ is asymptotically normally distributed with mean f_τ and variance $\tau(1-\tau)/(n[g(f_\tau)]^2)$ where

$$g(y) = y \sum_{i=1}^{n(y)} |f^{'}(z_i)|^{-1},$$

and $\{z_i\}$ denote those points in the sample space of X such that $f(z_i) = y, i = 1, 2, \cdots, n(y)$. Figure 2 (first row, right) depicts the confidence regions obtained with package **HDiR** (in dark red colour) for the circular model presented in Figure 1.

Suitable bandwidth selection for HDRs estimation

The plug-in reconstruction of the directional HDRs in (3) also involves the calculation of the kernel density estimator in (4) that is known to be heavily dependent on the selection of h. Package **HDiR** implements the proposal in Saavedra-Nieves and Crujeiras (2021b) where the first selector of h specifically designed for HDRs reconstruction is presented. The idea is to use an error criterion that quantifies the differences between the theoretical region and its plug-in reconstruction. In the real-valued setting, Samworth and Wand (2010) use a similar idea in order to propose one of the first bandwidth selectors for HDRs estimation.

The closed expression of the Hausdorff distance between the boundaries of the HDR and its plug-in reconstruction, $d_H(\partial L(f_\tau), \partial \hat{L}(\hat{f}_\tau))$, is not known in the directional case. However, such a distance could be approximated through a bootstrap procedure. With this view in mind, Saavedra-Nieves and Crujeiras (2021b) consider a new bandwidth selector as follows:

$$h_* = \arg\min_{h>0} \mathbb{E}_B \left[d_H(\partial L^*(\hat{f}_\tau^*), \partial \hat{L}(\hat{f}_\tau)) \right], \tag{5}$$

where $\mathbb{E}_B$ denotes the bootstrap expectation with respect to random samples of points $\mathcal{X}_n = \{X_1^*, \cdots, X_n^*\}$ generated from the directional kernel f_n that, of course, requires a pilot bandwidth chosen for computing $\hat{L}(\hat{f}_\tau)$.

3 Using HDiR

This section presents an overview of the structure of the package. **HDiR** (Saavedra-Nieves and Crujeiras, 2021a) is an easy-to-use toolbox that R practitioners can use for computation or plug-in estimation of directional highest density regions and general level sets defined on the circle and sphere. The methods included in the package facilitate both data exploration and nonparametric estimation of

the target regions. Functions in this library automatize the required operations for the computation of this kind of sets. First, we will describe the real data sets included in the package. Then, the functions available in **HDiR** are detailed. Of course, there exist several libraries in the CRAN repository of R dealing with plug-in estimation of Euclidean level sets and HDRs. In particular, the library **pdfCluster** (Azzalini et al., 2014) provides a routine to estimate the probability density function by kernel methods from a set of linear data with arbitrary dimension. The main focus is on cluster analysis via kernel density estimation according to the approach by Hartigan (1975). For modal clustering, package **LPCM** (Einbeck and Evers, 2019) implements the mean-shift algorithm and **Modalclust** (Cheng and Ray, 2014) performs the method for mode seeking introduced in Li et al. (2007). There are also other packages that do not solve the task of estimate HDRs directly, but they usually allow to compute the linear kernel density estimator and, therefore, address HDRs graphical representation (not necessarily with an appropriate estimate). A brief summary of the capabilities of these libraries are provided below.

- **denpro** (Klemelä, 2005, Klemelä, 2006, Klemelä, 2008, Klemelä, 2009, Holmström et al., 2017 and Klemelä, 2015): This library allows to visualize multivariate densities and density estimates with level set trees and also to represent level sets with shape trees in moderate dimensional cases. Furthermore, the kernel estimator implemented by default could be replaced by other density estimates.
- **hdrcde** (Hyndman et al., 2018): This package computes Euclidean HDRs in one and two dimensions. The specific HDR bandwidth selector proposed in Samworth and Wand (2010) is also implemented. Confidence regions for one-dimensional HDRs and bivariate HDRs scatterplots (colouring sample points according to the region in which they fall) are also available.
- **lsbs** (Doss and Weng, 2018): This package implements the bandwidth selector for two-dimensional Euclidean level sets and HDRs proposed in Doss and Weng (2018). A plug-in strategy to estimate the asymptotic risk function and minimize to get the optimal bandwidth matrix is applied.

Other packages such as **sm** (Bowman and Azzalini, 2018) and **ks** (Duong, 2007) also include tools for kernel density estimation allowing for graphical displays of density contours in the two- and three-dimensional Euclidean spaces. Moreover, there are many libraries in the CRAN repository for directional data analysis but, as far as we know, none of them solves the problem of level set or HDR reconstruction. In this section, we would like to highlight those packages including tools for kernel density estimation, both for circular and directional data:

- **circular** (Agostinelli and Lund, 2013): It is an extension of the **CircStats** package. It provides functions for the statistical analysis (descriptive statistics, circular models, hypothesis tests), graphical representation and some classical circular datasets.
- **Directional** (Tsagris et al., 2017): A collection of functions for directional data analysis are implemented in this library. Apart from hypothesis testing, discriminant and regression analysis, it allows to compute the kernel density estimation for hyper-spherical data using a von Mises-Fisher kernel.
- **DirStats** (García-Portugués, 2021): This library also allows to compute a kernel density estimator and, additionally, it implements the cross-validation and plug-in bandwidth selectors in Hall et al. (1987) and García-Portugués (2013), respectively.
- **NPCirc** (Oliveira et al., 2014): Nonparametric density and regression estimation methods for circular data are included in this package. Specifically, a circular kernel density estimation procedure is provided, jointly with different alternatives for choosing the smoothing parameter. Based on the kernel density estimator, a SiZer technique (CircSiZer) is developed for circular data. The package also includes functions for nonparametric circular regression.

Note also that there are other packages including tools for circular/directional data analysis. For instance, **CircStats** (Lund and Agostinelli, 2012) is a companion to Jammalamadaka and Sengupta (2001), although functions implemented in this package are also available in **circular**. **CircNNTSR** (Fernández-Durán and Gregorio-Domínguez, 2013) provides an alternative estimation method for circular distributions based on nonnegative trigonometric sums. **isocir** (Barragán et al., 2013) implements some routines for analyzing angular data subjected to order constraints on a unit circle. Finally, **movMF** (Hornik and Grün, 2014) is focused on mixtures of von Mises distributions, allowing to draw random samples from these models and to proceed with parameter estimation, by using an expectation-maximization algorithm.

Specifically, the goal of **HDiR** package is to provide tools for directional (circular and spherical) general level sets and HDRs exact computation also including their plug-in estimation. This library implements the first specific bandwidth selector devised for directional HDRs proposed in Saavedra-Nieves and Crujeiras (2021b), but it also allows directly user-defined bandwidth selection and to use the

Dataset	Description
`earthquakes`	Geographical coordinates (latitude and longitude) of earthquakes of magnitude greater than or equal to 2.5 degrees between October 2004 and April 2020
`sandhoppers`	Orientation of two sandhoppers species, Talitrus saltator and Talorchestia brito under different natural conditions
Function	**Description**
`circ.boot.bw`	Circular bootstrap bandwidth for HDRs estimation
`circ.distances`	Euclidean and Hausdorff distances between two sets of points on the unit circle
`circ.hdr`	Computation of HDRs and general level sets for a given circular real-valued function
`circ.plugin.hdr`	Circular plug-in estimation of HDRs and level sets and confidence regions
`circ.scatterplot`	Circular scatterplot for plug-in HDRs
`dspheremix`	Density functions for mixtures of spherical von Mises-Fisher
`rspheremix`	Random generation functions for mixtures of spherical von Mises-Fisher
`sphere.boot.bw`	Spherical bootstrap bandwidth for HDRs estimation
`sphere.distances`	Euclidean and Hausdorff distances between two sets of points on the unit sphere
`sphere.hdr`	Computation of HDRs and general level sets for a given spherical real-valued density
`sphere.plugin.hdr`	Spherical plug-in estimation of HDRs and level sets
`sphere.scatterplot`	Spherical scatterplot for plug-in HDRs

Table 1: Summary of **HDiR** package contents.

existing directional bandwidth selection methods devised for kernel density estimation. Additionally, two alternative methods for estimating the threshold f_τ (based on the proposal in Hyndman, 1996 and numerical integration methods, respectively) are developed. Moreover, confidence regions for circular HDR are also available and can be depicted for illustration. Two exploratory tools are also implemented. The first one is a scatterplot computed from HDRs plug-in reconstructions. Sample points are coloured according to the directional HDRs in which they fall. Finally, Euclidean and Hausdorff distances between sets can be also computed. Their roles are crucial to measure the distances between directional clusters or, for instance, to quantify the estimation error between the theoretical HDRs and the corresponding plug-in estimators.

A complete description of the **HDiR** package capabilities is provided in this section. The complete list of functions, illustrative density models (density functions and random sample generation) and the two novel datasets available in **HDiR**, with a brief description, can be seen in Table 1.

Data description

The package **HDiR** includes a circular and a spherical datasets, used for the illustration of the different functions. The first dataset, `sandhoppers`, contains the orientation angles (in radians between 0 and 2π) of two species of sandhoppers, Talitrus saltator and Talorchestia brito. Orientation was measured under natural conditions on the exposed nontidal sand of Zouara beach located in the Tunisian northwestern coast. Additionally, other variables of interest for analyzing the behavioral plasticity of both species were also registered. For instance, information on the month, the time of the day, the temperature, the air relative humidity or the sex of each animal is also available. This dataset was already analyzed in Scapini et al. (2002) and Marchetti and Scapini (2003). Specifically, the behavior of these two species is compared through regresion procedures. Scapini et al. (2002) conclude that Talitrus saltator showed more differentiated orientations, depending on the time of day, period of the year and sex, with respect to Talorchestia brito. As an illustration, Saavedra-Nieves and Crujeiras (2021b) also study the behavior of these two species of sandhoppers under the HDR estimation approach.

The second dataset, `earthquakes`, contains the geographical coordinates (latitude and longitude) of earthquakes of magnitude greater than or equal to 2.5 degrees on the Richter scale registered on Earth between 1st October 2004 and 9th April 2020. It can be downloaded from the website of the

European-Mediterranean Seismological Centre (EMSC)[1]. The planar points included in the dataset correspond to spherical coordinates on Earth. Due to the important damages that earthquakes of a certain intensity may cause, cluster detection of HDRs could be also useful to identify, from a real dataset, where earthquakes are specially likely. This information is crucial for decision-making, for example, to update construction codes guaranteeing a better building seismic-resistance. Saavedra-Nieves and Crujeiras (2021b) also analyze the recent world earthquakes distribution through HDRs estimation from this dataset. Results shows that the greatest mode of sample distribution is identified in the Southeast of Europe. Countries such as Italy, Greece or Turkey (located within this cluster) are, as expected, the most affected areas in the analyzed period. The second dataset, `earthquakes`, contains the geographical coordinates (latitude and longitude) of earthquakes of magnitude greater than or equal to 2.5 degrees on the Richter scale registered on Earth between 1st October 2004 and 9th April 2020. It can be downloaded from the website of the European-Mediterranean Seismological Centre (EMSC)[2]. The planar points included in the dataset correspond to spherical coordinates on Earth. Due to the important damages that earthquakes of a certain intensity may cause, cluster detection of HDRs could be also useful to identify, from a real dataset, where earthquakes are specially likely. This information is crucial for decision-making, for example, to update construction codes guaranteeing a better building seismic-resistance. Saavedra-Nieves and Crujeiras (2021b) also analyze the recent world earthquakes distribution through HDRs estimation from this dataset. Results shows that the greatest mode of sample distribution is identified in the Southeast of Europe. Countries such as Italy, Greece or Turkey (located within this cluster) are, as expected, the most affected areas in the analyzed period.

Spherical density models

Functions `dspheremix` and `rspheremix` allow to compute density functions and to generate data from the spherical distributions introduced in Saavedra-Nieves and Crujeiras (2021b). These densities represent a variety of complex structures showing multimodality and/or asymetry. Any user of package **HDiR** could use them for simulations or even for illustration purposes.

Function `dspheremix` computes the density function of 9 different spherical distributions that can be written as finite mixtures of spherical von Mises-Fisher. Function `rspheremix` is designed for random data generation from these 9 spherical models. Both functions have an argument called `model` which allows to specify a model (a number between 1 and 9) among the ones considered in Saavedra-Nieves and Crujeiras (2021b). The other inputs of `dspheremix` and `rspheremix` are `x` and `n`, respectively. `x` represents a matrix whose rows collect to points on the unit sphere (in Cartesian coordinates) and `n` denotes the number of observations to be randomly generated.

Specifically, model number 9 corresponds to the spherical density shown in Figure 1. For instance, the evaluation of this density on the north pole $(0, 0, 1)$ and the south pole $(0, 0, -1)$ can be easily obtained by:

```
> data <- rbind(c(1, 0, 0), c(0, 0, 1))
> dspheremix(x = data, model = 9)
[1] 0.0009079986 7.0233299246
```

Output of this example with `dspheremix` is a numeric vector containing the density values on both poles. Additionally, 100 random deviates from the same model can be obtained, fixing `set.seed(1)` as in the rest of examples throughout this work, by:

```
> rspheremix(n = 100, model = 9)
            [,1]         [,2]         [,3]
  [1,]  0.254793394 -0.186993591  0.948743233
  [2,]  0.227755936  0.896600223  0.379783194
  [3,] -0.227024808  0.516581111  0.825592934
  [4,]  0.125075316  0.960536966 -0.248444967
```

Output of function `rspheremix` is a matrix of dimension $n \times 3$ where each row corresponds to the Cartesian coordinates of a point generated on the unit sphere. For this example, the output is partially shown (only four of one hundred sample points are printed).

Computation of HDRs and general level sets with HDiR

Functions `circ.hdr` and `sphere.hdr` must be considered when the objective is to compute theoretical density level sets or HDRs from a fully known circular and spherical density f, respectively. However,

[1] European-Mediterranean Seismological Centre: www.emsc-csem.org.
[2] European-Mediterranean Seismological Centre: www.emsc-csem.org.

they could be also used for exact computation or plug-in estimation of general level sets when f is any (circular or spherical) real-valued function. In particular, level sets of a theoretical regression curve could be determined.

The basic arguments of function `circ.hdr` that the user must provide are the circular (not necessarily a density) function `f` and, depending on the set to be computed (a level set or a HDR), `level` or `tau` must be indicated. It is worth to mention that `level` represents the value of t in (1) and `1-tau`, the probability coverage required for HDR computation in (2). Note that `tau` must be specified only when `f` is a density. Otherwise, fixing the probability content of the level set makes no sense. Additionally, a graphical display is generated with different plot arguments (`col`, `lty`, `shrink`, $\cdots$). If no graphical representation is required, it is enough to consider `plot.hdr=FALSE` (by default `plot.hdr=TRUE`).

If `level` is specified, the output is a list with two components: `levelset`, a matrix where each row contains the boundaries (in radians) of each connected component of the level set and `level`, the input `level` or a character indicating if the level set is equal to the empty set or the support distribution. If `tau` is provided, the output is also a list with the next components: `hdr`, a matrix where each row contains the boundaries (in radians) of each connected component of the HDR; `prob.content`, probability coverage `1-tau` and `level`, threshold of the HDR computed by numerical integration methods.

An example with the code lines in order to computing a level set (second code line) and a HDR (third code line) for the circular density represented in Figure 1 is given below. This circular density is the model 13 implemented in the package **NPCirc**. Therefore, it is necessary to install this library before executing the following code.

```
> f <- function(x){return(dcircmix(x, 13))}
> circ.hdr(f, level = 0.35)
$levelset
          [,1]      [,2]
[1,] 0.3301974 0.6698291
[2,] 2.8271189 3.1730400
[3,] 4.9089351 5.0913298
$level
[1] 0.35
> circ.hdr(f, tau = 0.5)
$hdr
          [,1]      [,2]
[1,] 0.2232764 0.7767501
[2,] 2.7201978 3.2799611
[3,] 4.8523298 5.1479351
$prob.content
[1] 0.5
$level
[1] 0.3024789
```

From the outputs obtained, some conclusions on the number of connected components can be extracted. HDR computed when $\tau = 0.5$ has exactly three connected components with boundaries fully detailed in the element `hdr` of the obtained list. Density level set with threshold 0.35 is slightly different but the information in `levelset` also shows the existence of three connected components.

As for function `sphere.hdr`, argument `f` is now a spherical real-valued function. Again, `f` may not be a density. The other basic arguments `level`, `tau` and `plot.hdr` coincide with the usage description for function `circ.hdr`. Additionally, the user can specify two parameters related to the estimated boundary or to the numerical integration possibilities on the unit sphere to calculate the HDRs threshold. In particular, `nborder` indicates the maximum number of boundary points to be represented and `tol`, the tolerance parameter used to determinate the boundary. Two extra parameters control the numerical integration procedure, when required. Argument `mesh` indicates the number of vertices on each edge of the embedded icosaedrum (reproducing the meshes in Figure 3). Possible values of this argument are 10, 20 and 40, corresponding with 2000, 8000 and 32000 triangular cells on the sphere, respectively. Quadrature formulae on the triangles are possible with different degrees, controlled by `deg`, with values ranging from 0 up to 6.

An example with the code lines in order to compute a level set (second line) and a HDR (third line) for the spherical density represented in Figure 1 is presented in what follows:

```
> f <- function(x){return(dspheremix(x, model = 9))}
> sphere.hdr(f, level = 0.1, mesh = 10, deg = 3)
> sphere.hdr(f, tau = 0.5, mesh = 10, deg = 3)
```

Outputs are similar to those presented for function `circ.hdr`. Again, `levelset` and `hdr` are matrices of

rows of points (in Cartesian coordinates) on the level set and HDR boundaries, respectively. Moreover, it is worth to mention that execution time of `sphere.hdr` is considerably higher when `tau` is set instead `level` because, in this case, threshold estimation via numerical integration methods is required.

Plug-in estimation of HDRs and general level sets with HDiR

The **HDiR** package contains the implementation of density plug-in methods in order to estimate HDRs. Furthermore, it also enables plug-in estimation of general level sets.

Basic plug-in estimation of HDRs and density level sets

Function `circ.plugin.hdr` allows to reconstruct density level sets or HDRs from the kernel estimator described in (4). The arguments `tau`, `level` and `plot.hdr` have basically the same description for function `circ.hdr`. The argument `sample` denotes a numeric vector of angles (in radians) corresponding to the sample of points $\mathcal{X}_n$. The smoothing parameter to be used for kernel density estimation is denoted through `bw`. Its value could be directly established by the user. Following Oliveira et al. (2014), it could be also chosen by using the classical functions `bw.rt`, `bw.CV`, `bw.pi` or `bw.boot` in **NPCirc** (by default `bw=bw.CV(circular(sample))` providing a cross-validation bandwidth). The previous options are designed for density estimation. An appropriate bandwidth for HDR estimation can be obtained using `circ.boot.bw`. The argument `tau.method` is a character value selecting the rule to estimate the HDRs threshold. This must be one of `"quantile"` or `"trapezoidal"`. The default option estimates the threshold using the quantile method proposed in Hyndman (1996); the second one, using the trapezoidal rule for numerical integration. The confidence for limits on HDR are established from `conf` that is a numeric probability that takes the value `conf=0.95` by default. Finally, `plot.hdrconf` is a logical string. If `plot.hdr=TRUE` and `plot.hdrconf=TRUE` (default options), the confidence region for the estimated HDR is added to the estimation graphical representation. The argument `boot` is a logical string. If `TRUE`, confidence regions are not computed. Its name is due to this option is only used by function `circ.boot.bw` for reducing the execution time. Default `boot=FALSE`.

If `level` is specified, the output is a list with four components: `levelset`, a matrix where each row contains the boundary (in radians) of a connected component of the level set or a character indicating if the HDR is equal to the empty set or the support distribution; `prob.content`, the empirical probability coverage of the set; `level` indicates the level of the level set and `bw`, the value of the smoothing parameter. If `tau` is provided, the output is a list with the next components: `hdr`, a matrix where each row contains the boundary (in radians) of a connected component of the level set; `prob.content`, the probability coverage `1-tau`; `level`, the estimated threshold; `bw`, the numeric value of the smoothing parameter used; `hdr.lo` and `hdr.hi`, HDRs corresponding to lower and upper confidence limits, respectively; `threshold.lo` and `threshold.hi` the corresponding thresholds.

For example, the circular confidence regions in Figure 2 can be obtained from the next code lines:

```
> sample <- rcircmix(500, 13)
> circ.plugin.hdr(sample, tau = 0.5, plot.hdrconf = TRUE, k = 2, col = "blue")
$hdr
          [,1]      [,2]
[1,] 0.1478027 0.6761185
[2,] 2.6761715 3.2736716
[3,] 4.9403824 5.1542246
$prob.content
[1] 0.5
$level
      50%
0.2952482
$bw
[1] 64.62809
$hdr.lo
          [,1]      [,2]
[1,] 0.1226448 0.7327238
[2,] 2.6447241 3.3114085
[3,] 4.9089351 5.1793825
$level.lo
      50%
0.2762859
$hdr.hi
         [,1]      [,2]
```

```
[1,] 0.179250 0.6320922
[2,] 2.713908 3.2422243
```

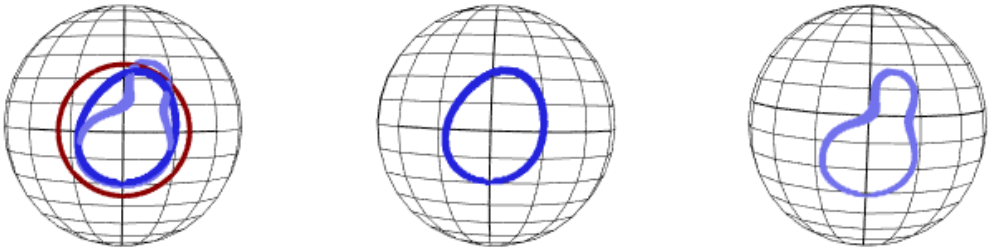


Figure 4: Theoretical (dark red colour) and estimated HDRs (left, bluish colours) from a random sample of size 500 when $\tau = 0.8$ using a cross-validation bandwidth and the specific bandwidth for spherical HDR reconstruction (left). Estimated HDRs from a random sample of size 500 using the specific bandwidth for spherical HDR reconstruction (center) and a cross-validation bandwidth (right) when $\tau = 0.8$.

```
[3,] 4.984409 5.1164877
$level.hi
      50%
0.3142105
```

Specifically, `hdr.lo` and `hdr.hi` in the output list contain the matrices whose rows correspond to the boundaries (in radians) of the connected components of lower and upper confidence regions, respectively. For this example, both regions have three connected components. Additionally, `level.lo` and `level.hi` contain the thresholds of both confidence sets.

The specific bandwidth for circular HDRs estimation described in Saavedra-Nieves and Crujeiras (2021b) can be computed from function `circ.boot.bw`. As in the previous circular functions described, the argument `sample` is a numeric vector of angles (in radians) representing $\mathcal{X}_n$ and `tau` corresponds to the probability coverage `1-tau` of the HDR to be reconstructed. The pilot smoothing parameter used is `bw`. Default `bw=bw.CV(circular(sample),upper = 100)`. As before, its value could be chosen by using the classical functions `bw.rt`, `bw.CV`, `bw.pi` or `bw.boot` in **NPCirc**. The number of bootstrap resamples is denoted by `B` (by default `B=50`) and `upper` is the numerical upper value for bounding the optimization procedure (by default `1.5bw`). The output of this function is a single numeric value corresponding to the selected smoothing parameter.

The following code lines show how to determine both bandwidths for the circular sample previously generated. Output shows that cross-validation selector takes a larger value than the proposal in Saavedra-Nieves and Crujeiras (2021b).

```
> bw.CV(sample, upper = 100); circ.boot.bw(sample, tau = 0.8, B = 2)
[1] 64.62809
[1] 37.06194
```

Function `sphere.plugin.hdr` is designed to estimate spherical HDRs or density level sets from the kernel estimator described in (4). The arguments `tau`, `level`, `plot.hdr`, `nborder`, `tol`, `mesh` and `deg` have the same description as for function `sphere.hdr`. The pilot smoothing parameter used is `bw` that, by default, is `bw="none"` selecting a cross-validation bandwidth. Although other options are possible. For instance, `bw` can be a numeric value o also `bw="rot"` allows to consider the rule of thumb suggested by García-Portugués (2013). The value of `bw` could be also selected directly by the user. The argument `ngrid` sets the resolution of the density calculation (by default `ngrid=500`).

If `level` is provided, the output is also a list with four components: `levelset`, a matrix of rows of points (on the HDR boundary; `prob.content`, the empirical probability coverage of the set `1-tau`; `level`, the level of the HDR and `bw`, the value of the smoothing parameter. If `tau` is an input, the output of `sphere.plugin.hdr` is a list with the following components: `hdr`, a matrix of rows of points on the HDR boundary; `prob.content`, probability coverage `1-tau` and `level`, threshold or level associated to the probability content `1-tau`. The threshold f_τ is computed through the algorithm proposed in Hyndman (1996). Numerical integration is not considered here in order to reduce the computation time.

The spherical HDRs estimators in Figure 2 can be reproduced through the next code lines:

```
> sample <- rspheremix(500, model = 9)
> sphere.plugin.hdr(sample, tau = 0.5, nborder = 2000)
```

The first specific bandwidth for spherical HDRs estimation described in Saavedra-Nieves and

Crujeiras (2021b) can be computed from function `sphere.boot.bw`. As in the previous spherical functions described, the argument `sample` is a matrix whose rows represent points on the unit sphere (in Cartesian coordinates) and `tau` corresponds to the probability coverage `1-tau` of the HDR to be reconstructed. The pilot smoothing parameter `bw` (default `bw="none"`) is chosen using cross-validation, although it may be set to a numeric value or `bw="rot"`, allowing to select the rule of thumb suggested by García-Portugués (2013). The argument `B` denotes again the number of bootstrap resamples that (default `B=50`) and `upper` is the numerical upper value for bounding the optimization procedure (default `1.5bw`). The output of this function is a single numeric value corresponding to the selected smoothing parameter.

The following code lines contain a simulated example where the cross-validation bandwidth and the proposal in Saavedra-Nieves and Crujeiras (2021b) provide HDR estimations which look quite different for the spherical model 8 in **HDiR**. Figure 4 shows the graphical representations of the theoretical HDR to be estimated when $\tau = 0.8$ (dark red colour) and the corresponding reconstructions (bluish colours) obtained from a random sample of size 500. In this case, the specific bandwidth for spherical HDRs reconstruction takes the value 0.28 while the cross-validation bandwidth is equal to 0.20.

```
> sample <- rspheremix(500, model = 8)
> bw.boot <- sphere.boot.bw(sample, bw = "rot", tau = 0.8, B = 2)
> sphere.plugin.hdr(sample, bw = bw.boot, tau = 0.8)
> sphere.plugin.hdr(sample, bw = "none", tau = 0.8)
```

Finally, it is important to note that function `sphere.plugin.hdr` for reconstructing spherical HDR's calls `vmf.kerncontour` in package **Directional** to compute the density on a grid on the sphere. Most of the computational work in this function is in estimating the density using `vmf.kerncontour`. Hence, the speed of this function depends largely on the speed of `vmf.kerncontour`. A similar situation occurs for function `sphere.boot.bw` where function `sphere.plugin.hdr` is called $(B+1)$ times where B indicates the number of bootstrap resamples.

Plug-in estimation of HDRs and density level sets from an arbitrary density estimator

Density estimators different from the one introduced in (4) could be naturally considered for plug-in estimation of HDRs or level sets. Functions `circ.hdr` and `sphere.hdr` in package **HDiR** allow to consider this option in the circular and spherical settings, respectively.

Next, an example with the code lines in order to determine a spherical HDR plug-in reconstruction (sixth line) from the kernel density estimator in Bai et al. (1989) with uniform kernel is shown.

```
> f <- function(x){
    sample <- rspheremix(500, model = 3)
    return(kde_dir(x, data = sample, h = 0.4,
     L = function(x) dunif(x)))
    }
> sphere.hdr(f, level = 0.3)
$levelset
               [,1]         [,2]      [,3]
  [1,]  0.3587511132 -0.159961736 0.9196249
  [2,] -0.4523490796  0.077542650 0.8884635
  [3,] -0.4588831000  0.060463844 0.8864369
  [4,]  0.2455354599 -0.291602658 0.9244892
$level
[1] 0.3
```

An spherical density estimator with uniform kernel is available in package **DirStats**. Before level set plug-in estimation, it is necessary to install this library in order to define the kernel estimator, in this example, from a sample of size 500 of model 3 in **HDiR** (lines from 1 to 5). The output contains a matrix of points on the boundary of the plug-in estimator in `levelset`. Note that only the first four points are printed in the example. The value of the threshold 0.3 considered for reconstruction is also shown in `level`.

Furthermore, if the considered density estimator for plug-in estimation is also a density function, argument `tau` in `circ.hdr` and `sphere.hdr` could be used.

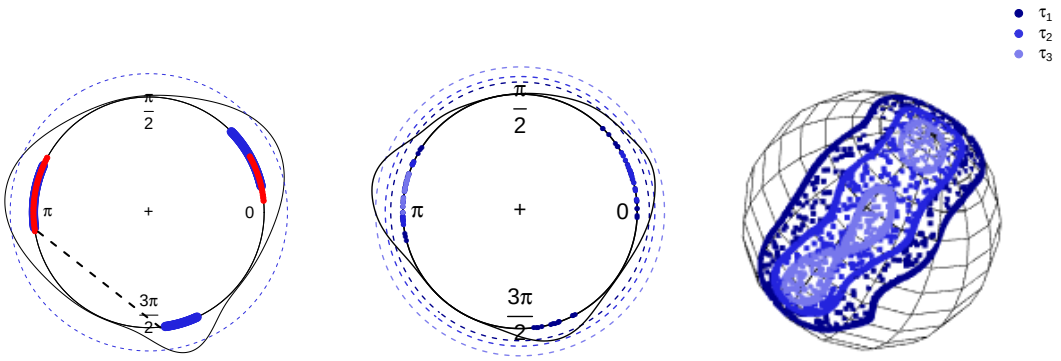


Figure 5: In the first column, the black dotted line represents the Hausdorff distance between $\partial L(f_\tau)$ (blue colour) and $\partial \hat{L}(\hat{f}_\tau)$ (red colour) for the circular density shown in Figure 1 when $\tau = 0.5$. In the second and third columns, scatterplots showing $\hat{L}(\hat{f}_{\tau_i})$ $(i = 1, 2, 3)$ for the circular and spherical densities contained in Figure 1 when $\tau_1 = 0.2$, $\tau_2 = 0.5$ and $\tau_3 = 0.8$. The circular scatterplot was computed from $\mathcal{X}_{100}$ and the spherical scatterplot from $\mathcal{X}_{1000}$.

Plug-in estimation of general level sets

A generalisation of the approach in Cuevas et al. (2006), for general level sets, to the directional setting can be performed in practice with **HDiR**. Again, functions `circ.hdr` and `sphere.hdr` address this problem for circular and sperical data, respectively.

Next, an example with the code lines in order to obtain the plug-in estimator of a regression curve (eighth line) with circular explanatory (`x`) variable and linear response (`y`). In this case, the regression curve is estimated through the Nadaraya-Watson estimator implemented in **NPCirc**. Here, it is computed from a sample of size 100 of variables `x` and `y` (lines from 1 to 7).

```
> f <- function(t){
    n <- 100
    x <- runif(n, 0, 2*pi)
    y <- sin(x)+0.5*rnorm(n)
    return(kern.reg.circ.lin(circular(x), y, t, bw = 10, method = "NW")$y)
    }
> circ.hdr(f, level = 0.5, plot.hdr = FALSE)
$levelset
          [,1]     [,2]
[1,] 0.4748553 2.757935
$level
[1] 0.5
```

Output in `levelset` contains the boundary (in radians) of the only connected component for the reconstructed regression level set.

Exploring data with HDiR

This section introduces a brief background on the design of two exploratory tools included in **HDiR**: distances between sets and circular/spherical scatterplots.

Distances between sets are a useful tool when the target is the reconstruction of a set. In particular, the Hausdorff distance can be seen as a suitable error criterion also in the directional setting. Additionally, it could be also used for measuring the distances between modes or clusters of two different populations. Figure 5 (first column) represents, through a black dashed line, the Hausdorff distance between $\partial L(f_\tau)$ (blue colour) and $\partial \hat{L}(\hat{f}_\tau)$ (red colour) for the circular density shown in Figure 1 when $\tau = 0.5$. Note that the maximum value of this error criterion is 2, the diameter of the unit circle. In this example, the Hausdorff estimation error that is equal to 1.38 is remarkably high.

Function `circ.distances` computes the Euclidean and Hausdorff distances between two sets of points in S^1. Its inputs are `x` and `y`, two numeric vectors of angles (in radians) determining both sets of points. The output is a list with two components: `dE`, a numeric value corresponding to the Euclidean distance, and `dH`, another numeric value corresponding to the Hausdorff distance.

Specifically, if x and y correspond to two HDRs boundaries, this function returns the distances between the circular HDRs frontiers. In particular, for the example in Figure 5 (left), the distances between $\partial L(\tau)$ and $\partial \hat{L}(\hat{f}_\tau)$ can be computed from the next code lines:

```
> sample <- rcircmix(100, 13)
> f <- function(x){return(dcircmix(x, 13))}
> circ.distances(as.numeric(circ.hdr(f, tau = 0.5)$hdr),
    + as.numeric(circ.plugin.hdr(sample, tau = 0.5)$hdr))
$dE
[1] 0.04402277
$dH
[1] 1.37933
```

The results obtained show that the Euclidean distance is considerably smaller than the Hausdorff distance that, as we mention before, takes the value 1.38.

Function `sphere.distances` also determines the Euclidean and Hausdorff distances but, in this case, between two sets of points on S^2. Now, the inputs x and y are two matrices whose rows represent points on the unit sphere (in Cartesian coordinates). The output of this function has the same organization as the output of `circ.distances` and it also allows to compute distances between spherical HDRs frontiers.

Distances between $\partial \hat{L}(\hat{f}_{\tau_2})$ and $\partial \hat{L}(\hat{f}_{\tau_3})$ represented in Figure 5 (right) can be computed from the next code lines:

```
> sample = rspheremix(1000, model = 9)
> x <- sphere.plugin.hdr(sample, tau = 0.8, plot.hdr = FALSE)$hdr
> y <- sphere.plugin.hdr(sample, tau = 0.5, plot.hdr = FALSE)$hdr
> sphere.distances(x, y)
$dE
[1] 0.08600028
$dH
[1] 0.258705
```

The performance of the specific bandwidth for HDR estimation introduced in Saavedra-Nieves and Crujeiras (2021b) can be also illustrated through the consideration of the Hausdorff distance in the example shown in Figure 4. Specifically, the value of the Hausdorff distance between the theoretical HDR and the reconstruction computed from the bandwidth proposed in Saavedra-Nieves and Crujeiras (2021b) is 0.20. However, the Hausdorff distance increases considerably, taking the value 0.36, when it measures the discrepancies between the theoretical HDR and the corresponding estimator obtained from a cross-validation approach.

Additionally, scatterplots are useful to identify the estimated directional HDRs in which sample points fall. This graphical tool is computed as follows. Given several values $\tau_1, \cdots, \tau_k \in (0,1)$ $(k \geq 1)$ and a random sample of points $\mathcal{X}_n$, the estimated HDRs $\hat{L}(\hat{f}_{\tau_1}), \cdots, \hat{L}(\hat{f}_{\tau_k})$ are represented using different colours jointly with the subset of sample points belonging to each of them. Figure 5 (second and third columns) displays the scatterplots for $\tau_1 = 0.2$, $\tau_2 = 0.5$ and $\tau_3 = 0.8$ for the circular and the spherical densities shown in Figure 1. They were calculated from random samples of sizes $n = 100$ and $n = 1000$, respectively.

Function `circ.scatterplot` produces a circular scatterplot with points coloured according to the HDRs in which they fall. Apart from the argument `tau` that represents a numeric vector of probabilities and `plot.density` that is a logical string indicating if the kernel density estimator is added to the scatterplot (default `plot.density=TRUE`), the other inputs (`sample`, `bw` and `tau.method`) have the same description for circular functions. The output is a scatterplot and also a list where the number of components is equal to the number of estimated HDR or, equivalently, to the length of `tau` vector. Each component contains the sample points in each HDR from the smallest value of `tau` to the largest one.

Next code lines allow to obtain a circular scatterplot computed from a circular sample of size 100 as the shown in Figure 5 (second column).

```
> sample<- rcircmix(100, model = 13)
> circ.scatterplot(sample, tau = c(0.2, 0.5, 0.8))
```

Spherical scatterplots can be represented from function `sphere.scatterplot`. Again, apart from `tau` that is a vector of probabilities, the description of the remaining parameters coincides with the rest of spherical functions. The output provides a scatterplot and, as in the circular case, a list where the number of components is equal to the number of estimated HDR containing the corresponding sample points from the smallest value of `tau` to the biggest one.

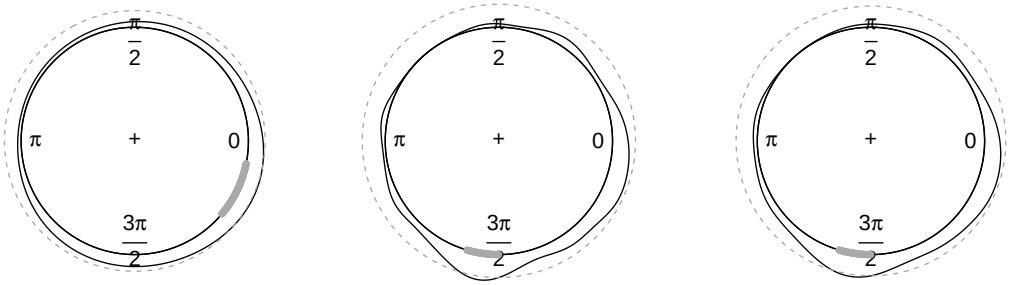


Figure 6: Plug-in estimations of HDRs (gray colour) with cross-validation bandwidth, when $\tau = 0.8$, for females (left) and males (center) of the species Talorchestia Brito when the orientation is registered in morning during October. Plug-in estimation of HDR (gray colour) with specific bandwidth h_*, when $\tau = 0.8$, for males (right) of the species Talorchestia Brito when the orientation is registered in morning during October.

As an illustration, the spherical scatterplot shown in Figure 5 (third column) could be computed from the next code lines:

```
> sample <- rspheremix(1000, model = 9)
> sphere.scatterplot(sample, tau = c(0.2, 0.5, 0.8))
```

Real data analysis with HDiR

Datasets `sandhoppers` and `earthquakes` included in **HDiR** are used next to illustrate briefly the usage of the set of functions previously described in the circular and spherical settings, respectively.

Figure 6 shows the estimated HDRs established in (3), when $\tau = 0.8$, for female (left) and male sandhoppers (right) of the species Talorchestia Brito when the orientation is registered in the morning during October. The largest modes of both distributions are located in completely different directions, indicating that variable sex is a factor with influence on the sandhoppers behavior. The code lines used are presented:

```
> data(sandhoppers)
> attach(sandhoppers)
> britoF <- angle[(species == "brito")&(time == "morning")&(sex == "F")
        + &(month == "October")]
> circ.plugin.hdr(sample = britoF, tau = 0.8, plot.hdrconf = FALSE)
> britoM <- angle[(species == "brito")&(time == "morning")&(sex == "M")
    + &(month == "October")]
> circ.plugin.hdr(sample = britoM, tau = 0.8, plot.hdrconf = FALSE)
```

According to Figure 6, no remarkable differences exist between the HDRs reconstructions for males using a cross-validation bandwidth (center) and the proposal h_* in Saavedra-Nieves and Crujeiras (2021b) (right). However, these smoothing parameters are quite different, taking values 33.86 and 19.47, respectively. For the subset of females, differences between smoothing parameters are smaller (5.78 and 3.39, respectively). Next, code lines show how to determine both bandwidths for the group of males (fist line) and females (second line).

```
> bw.CV(britoM); circ.boot.bw(britoM, tau = 0.8)
> bw.CV(britoF); circ.boot.bw(britoF, tau = 0.8)
```

As an example with the dataset `earthquakes` in Figure 7, we show the estimated HDR defined in (3) for $\tau = 0.8$. The largest mode of the earthquakes distribution is located in Southeast Europe. Note that it is necessary to install the packages **Directional**, **ggplot2**, **maps** and **mapproj** previously to obtain this figure.

```
> data(earthquakes)
> hdr <- as.data.frame(euclid.inv(sphere.plugin.hdr(euclid(earthquakes), tau = 0.8,
        + plot.hdr = FALSE)$hdr))
> world <- map_data("world")
> g.earthquakes <- ggplot()+
> geom_map(data = world, map = world, mapping = aes(map_id = region),
        +  color = "grey90", fill = "grey80")+
```

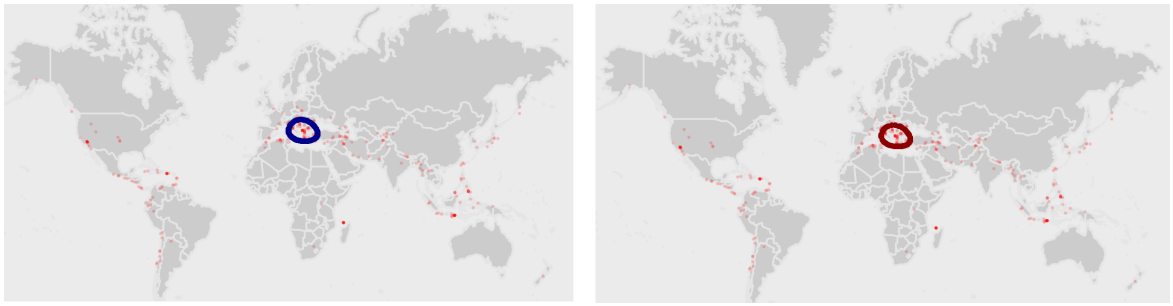


Figure 7: Contours of plug-in HDRs for $\tau = 0.8$ obtained from the sample of world earthquakes registered between October 2004 and April 2020 with cross-validation bandwidth (left) and with the specific bandwidth for spherical HDRs reconstruction (right).

```
> geom_point(data = earthquakes, mapping = aes(x = Longitude,
        +   y = Latitude), color = "red", alpha = 0.2, size = 0.75, stroke = 0)+
> geom_point(data = hdr, mapping = aes(x = Long, y = Lat),
        +          color = "darkblue", size = 1)+
> scale_y_continuous(breaks = NULL, limits = c(-90, 90))+
> scale_x_continuous(breaks = NULL, limits = c(-180, 180))+
> coord_map("mercator")
> g.earthquakes
```

The value of the bandwidth proposed in Saavedra-Nieves and Crujeiras (2021b) for `earthquakes` dataset with `tau=0.8` and `B=5` bootstrap resamples is 0.09 and it can be obtained from the next code line. In this particular case, Figure 7 shows that there is not a large differences between the HDRs reconstructed from cross-validation bandwidth (left) and the proposal in Saavedra-Nieves and Crujeiras (2021b) (right).

```
> sphere.boot.bw(euclid(earthquakes), tau = 0.8, B = 5)
```

Once the HDRs estimation has been performed for different values of τ, Euclidean and Hausdorff distances between the blue and red contours in Figure 7 are useful to analyse differences between them. For the previous example, distances can be computed from the following code lines. Note that the value of the bandwidth in Saavedra-Nieves and Crujeiras (2021b) has been directly inserted as an argument in the fourth line. Values obtained for Euclidean and Hausdorff distances are 0 and 0.02, respectively.

```
> hdr1 <- sphere.plugin.hdr(euclid(earthquakes), tau = 0.8, plot.hdr = FALSE)$hdr
> hdr2 <- sphere.plugin.hdr(euclid(earthquakes), bw = 0.09, tau = 0.8,
        + plot.hdr = FALSE)$hdr
> sphere.distances(hdr1, hdr2)
```

Apart from distances between HDRs, scatterplots are another powerful exploratory tool implemented in **HDiR**. For the `sandhoppers` dataset, Figure 8 shows the circular scatterplots for $\tau = 0.2, 0.5$ and 0.8 for females (left) and males (center) of the species Talorchestia Brito when the orientation is registered in the morning during October when $\tau = 0.2, 0.5$ and 0.8. They can be obtained from the following code:

```
> circ.scatterplot(britoF, tau = c(0.2, 0.5, 0.8))
> circ.scatterplot(britoM, tau = c(0.2, 0.5, 0.8))
```

Spherical scatterplots for `earthquakes` dataset when $\tau = 0.2$, $\tau = 0.5$ and $\tau = 0.8$ can be computed from the following code line. The function `euclid` allows to transforms the data to geographical coordinates (longitude and latitude) on Cartesian coordinates. Remark that the smoothing parameter is selected by using the rule of thumb proposed in García-Portugués (2013).

```
>  sphere.scatterplot(euclid(earthquakes), tau = c(0.2, 0.5, 0.8), bw = "rot",
    + nborder = 1500)
```

4 Discussion

HDiR has been mainly developed for facilitating the reconstruction of directional (circular and spherical) HDRs and density level sets, following a nonparametric plug-in approach. However, it

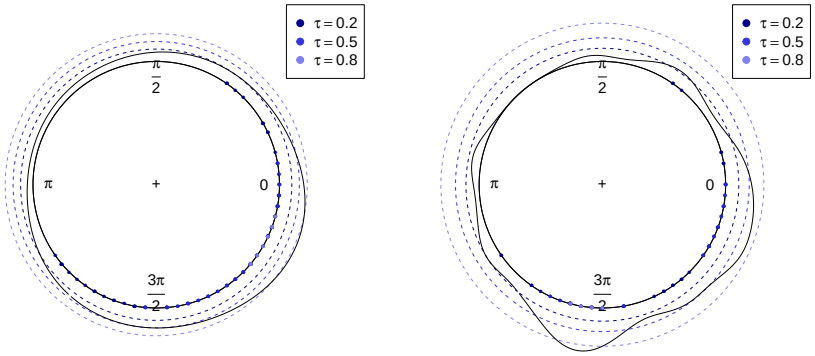


Figure 8: Circular scatterplots computed for $\tau = 0.2$, 0.5 and 0.8 from samples of females (left) and males (center) of the species Talorchestia Brito when the orientation is registered in morning during October.

also allows to solve the computation and the plug-in estimation of level sets for general real-valued functions, such as a regression curve. As consequence, plug-in reconstruction of HDRs could be performed by considering a different density estimator than the one implemented by default in **HDiR**.

The implemented tools are accessible for the scientific community, enabling their usage in practical problems such as the exploration of modes or the approximation of the distribution *effective support*. As previously noted, level set computation is also useful for determining distribution clusters, a task that can be accomplished by the identification of the connected components from a plug-in level set estimator.

Up to the authors' knowledge, **HDiR** is the only statistical package that allows to estimate (circular and spherical) HDRs and general level sets. For HDRs reconstruction, **HDiR** also implements the first specific selector for HDRs estimation in this context, proposed in Saavedra-Nieves and Crujeiras (2021b). Additionally, it offers graphical exploratory tools such as HDRs scatterplots that allow to visualize HDRs of a distribution taking into account different probability contents. Similarities or discrepancies between them could be measured through the Hausdorff distance also implemented in **HDiR**.

Future extensions of the **HDiR** package could include the estimation of level sets and HDRs in other supports, involving a circular or a spherical component, such as the torus or the cylinder. In addition, new specific bandwidths for HDR estimation could be implemented. A variety of bandwidths selectors emerge from the consideration of different distances in (5). Finally, cluster definition in Hartigan (1975) deserves to be exploited in the directional setting, for instance, by implementing cluster trees for hyperspherical data.

5 Acknowledgments

P. Saavedra-Nieves and R.M. Crujeiras acknowledge the financial support of Ministerio de Ciencia e Innovación of the Spanish government under grants PID2020-118101GB-I00 and PID2020-116587GB-I00 and ERDF. Authors also thank Prof. Felicita Scapini for providing the sandhoppers data (collected under the support of the European Project ERB ICI8-CT98-0270), Prof. Andrés Prieto for his help with spherical numerical integration. The authors also acknowlege the constructive comments of the AE and the reviewer, which have improved the contents of the paper and the package.

Bibliography

C. Agostinelli and U. Lund. R package 'circular': Circular statistics. 2013. URL `https://r-forge.r-project.org/projects/circular`. R package version 0.4-7. [p126]

A. Azzalini and N. Torelli. Clustering via nonparametric density estimation. *Statistics and Computing*, 17(1):71–80, 2007. [p121, 122]

A. Azzalini, G. Menardi, and T. Rosolin. Package pdfcluster: Cluster analysis via nonparametric density estimation. *J. Stat. Softw*, 11:1–26, 2014. [p126]

Z. Bai, C. R. Rao, and L. Zhao. Kernel estimators of density function of directional data. In *Multivariate Statistics and Probability*, pages 24–39. Elsevier, 1989. [p123, 133]

A. Baíllo. Total error in a plug-in estimator of level sets. *Statistics & probability letters*, 65(4):411–417, 2003. [p123]

A. Baíllo and A. Cuevas. Parametric versus nonparametric tolerance regions in detection problems. *Computational Statistics*, 21(3):523–536, 2006. [p122]

S. Barragán, M. A. Fernández, C. Rueda, and S. D. Peddada. isocir: An R package for constrained inference using isotonic regression for circular data, with an application to cell biology. *Journal of Statistical Software*, 54(4), 2013. [p126]

A. Bowman and A. Azzalini. Package 'sm'. 2018. URL https://CRAN.R-project.org/package=sm. R package version 2.2-5.6. [p126]

G. Box and G. Tiao. Bayesian inference in statistical analysis, reading. *Mass.: Addison-Wesley*, 1973. [p121]

A. Casa, J. E. Chacón, and G. Menardi. Modal clustering asymptotics with applications to bandwidth selection. *Electronic Journal of Statistics*, 14(1):835–856, 2020. [p122]

S.-J. Chang-Chien, M.-S. Yang, and W.-L. Hung. Mean shift-based clustering for directional data. In *Third International Workshop on Advanced Computational Intelligence*, pages 367–372. IEEE, 2010. [p121]

Y.-C. Chen, C. R. Genovese, and L. Wasserman. Density level sets: Asymptotics, inference, and visualization. *Journal of the American Statistical Association*, 112(520):1684–1696, 2017. [p123]

Y. Cheng and S. Ray. Parallel and hierarchical mode association clustering with an R package modalclust. *Open Journal of Statistics*, 4(10):826–836, 2014. [p126]

A. Cholaquidis, R. Fraiman, and L. Moreno. Level set and density estimation on manifolds. *Journal of Multivariate Analysis*, 189:104925, 2022. [p121, 123]

D. R. Cox and D. V. Hinkley. *Theoretical statistics*. CRC Press, 1979. [p125]

A. Cuevas, W. González-Manteiga, and A. Rodríguez-Casal. Plug-in estimation of general level sets. *Australian & New Zealand Journal of Statistics*, 48(1):7–19, 2006. [p121, 122, 123, 134]

M. Di Marzio, A. Panzera, and C. C. Taylor. Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6):2156–2173, 2011. [p123]

C. R. Doss and G. Weng. Bandwidth selection for kernel density estimators of multivariate level sets and highest density regions. *Electronic Journal of Statistics*, 12(2):4313–4376, 2018. [p126]

T. Duong. ks: Kernel density estimation and kernel discriminant analysis for multivariate data in R. *Journal of Statistical Software*, 21(7):1–16, 2007. [p126]

J. Einbeck and L. Evers. Lpcm: Local principal curve methods. 2019. URL https://CRAN.R-project.org/package=LPCM. R package version 0.46-3. [p126]

J. Fernández-Durán and M. Gregorio-Domínguez. Circnntsr: An R package for the statistical analysis of circular data using nonnegative trigonometric sums (nnts) models. 2013. URL https://CRAN.R-project.org/package=CircNNTSR. R package version 2.2-1. [p126]

E. García-Portugués. Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electronic Journal of Statistics*, 7:1655–1685, 2013. [p123, 126, 132, 133, 137]

E. García-Portugués. DirStats: Nonparametric methods for directional data. 2021. URL https://CRAN.R-project.org/package=DirStats. R package version 0.1.7. [p126]

P. Hall, G. Watson, and J. Cabrera. Kernel density estimation with spherical data. *Biometrika*, 74(4): 751–762, 1987. [p123, 126]

J. A. Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975. [p121, 126, 138]

L. Holmström, K. Karttunen, and J. Klemelä. Estimation of level set trees using adaptive partitions. *Computational Statistics*, 32(3):1139–1163, 2017. [p126]

K. Hornik and B. Grün. movmf: an R package for fitting mixtures of von mises-fisher distributions. *Journal of Statistical Software*, 58(10):1–31, 2014. [p126]

R. J. Hyndman. Computing and graphing highest density regions. *The American Statistician*, 50(2): 120–126, 1996. [p122, 125, 127, 130, 132]

R. J. Hyndman, J. Einbeck, and M. Wand. hdrcde. 2018. URL https://CRAN.R-project.org/package=hdrcde. R package version 3.4. [p126]

S. R. Jammalamadaka and A. Sengupta. *Topics in circular statistics*, volume 5. world scientific, 2001. [p126]

J. Klemelä. Algorithms for manipulation of level sets of nonparametric density estimates. *Computational Statistics*, 20(2):349–368, 2005. [p126]

J. Klemelä. Visualization of multivariate density estimates with shape trees. *Journal of Computational and Graphical Statistics*, 15(2):372–397, 2006. [p126]

J. Klemelä. Mode trees for multivariate data. *Journal of Computational and Graphical Statistics*, 17(4): 860–869, 2008. [p126]

J. Klemelä. denpro. 2015. URL https://CRAN.R-project.org/package=denpro. R package version 0.9.2. [p126]

J. S. Klemelä. *Smoothing of multivariate data: density estimation and visualization*. John Wiley & Sons, 2009. [p126]

J. Li, S. Ray, and B. G. Lindsay. A nonparametric statistical approach to clustering via mode identification. *Journal of Machine Learning Research*, 8(8), 2007. [p126]

U. Lund and C. Agostinelli. Circstats. 2012. URL https://CRAN.R-project.org/package=CircStats. R package version 0.2-6. [p126]

E. Mammen and W. Polonik. Confidence regions for level sets. *Journal of Multivariate Analysis*, 122: 202–214, 2013. [p123]

G. M. Marchetti and F. Scapini. Use of multiple regression models in the study of sandhopper orientation under natural conditions. *Estuarine, Coastal and Shelf Science*, 58:207–215, 2003. [p127]

D. M. Mason and W. Polonik. Asymptotic normality of plug-in level set estimates. *The Annals of Applied Probability*, 19(3):1108–1142, 2009. [p123]

G. Menardi. A review on modal clustering. *International Statistical Review*, 84(3):413–433, 2016. [p121]

M. Oliveira, R. M. Crujeiras, and A. Rodríguez-Casal. A plug-in rule for bandwidth selection in circular density estimation. *Computational Statistics & Data Analysis*, 56(12):3898–3908, 2012. [p123]

M. Oliveira, R. M. Crujeiras, and A. Rodríguez-Casal. Npcirc: An r package for nonparametric circular methods. *Journal of Statistical Software, Articles*, 61(9):1–26, 2014. [p126, 130]

B. Pelletier. Kernel density estimation on riemannian manifolds. *Statistics & probability letters*, 73(3): 297–304, 2005. [p123]

P. Rigollet, R. Vert, et al. Optimal rates for plug-in estimators of density level sets. *Bernoulli*, 15(4): 1154–1178, 2009. [p123]

P. Saavedra-Nieves and R. M. Crujeiras. HDiR. 2021a. URL https://CRAN.R-project.org/package=HDiR. R package version 1.1.2. [p125]

P. Saavedra-Nieves and R. M. Crujeiras. Nonparametric estimation of directional highest density regions. *Advances in Data Analysis and Classification*, pages 1–36, 2021b. [p122, 123, 124, 125, 126, 127, 128, 132, 133, 135, 136, 137, 138]

R. Samworth and M. Wand. Asymptotics and optimal bandwidth selection for highest density region estimation. *The Annals of Statistics*, 38(3):1767–1792, 2010. [p122, 125, 126]

F. Scapini, A. Aloia, M. F. Bouslama, L. Chelazzi, I. Colombini, M. ElGtari, M. Fallaci, and G. M. Marchetti. Multiple regression analysis of the sources of variation in orientation of two sympatric sandhoppers, talitrus saltator and talorchestia brito, from an exposed mediterranean beach. *Behavioral Ecology and Sociobiology*, 51(5):403–414, 2002. [p127]

G. Strang and G. J. Fix. *An analysis of the finite element method*. Prentice-hall, 1973. [p124]

C. C. Taylor. Automatic bandwidth selection for circular density estimation. *Computational Statistics & Data Analysis*, 52(7):3493–3500, 2008. [p123]

M. Tsagris, G. Athineou, and A. Sajib. Directional. 2017. URL https://CRAN.R-project.org/package=Directional. R package version 5.6. [p126]

A. B. Tsybakov. On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3): 948–969, 1997. [p123]

M.-S. Yang, S.-J. Chang-Chien, and H.-C. Kuo. On mean shift clustering for directional data on a hypersphere. In *International Conference on Artificial Intelligence and Soft Computing*, pages 809–818. Springer, 2014. [p121]

Paula Saavedra-Nieves
CITMAga, Galician Centre for Mathematical Research and Technology
Universidade de Santiago de Compostela
Facultade de Matemáticas
Lope Gómez de Marzoa, s/n
Campus sur, 15782
Santiago de Compostela, Spain
paula.saavedra@usc.es

Rosa M. Crujeiras
CITMAga, Galician Centre for Mathematical Research and Technology
Universidade de Santiago de Compostela
Facultade de Matemáticas
Lope Gómez de Marzoa, s/n
Campus sur, 15782
Santiago de Compostela, Spain
rosa.crujeiras@usc.es

metapack: An R Package for Bayesian Meta-Analysis and Network Meta-Analysis with a Unified Formula Interface

by Daeyoung Lim, Ming-Hui Chen, Joseph G. Ibrahim, Sungduk Kim, Arvind K. Shah and Jianxin Lin

Abstract Meta-analysis, a statistical procedure that compares, combines, and synthesizes research findings from multiple studies in a principled manner, has become popular in a variety of fields. Meta-analyses using study-level (or equivalently *aggregate*) data are of particular interest due to data availability and modeling flexibility. In this paper, we describe an R package **metapack** that introduces a unified formula interface for both meta-analysis and network meta-analysis. The user interface—and therefore the package—allows flexible variance-covariance modeling for multivariate meta-analysis models and univariate network meta-analysis models. Complicated computing for these models has prevented their widespread adoption. The package also provides functions to generate relevant plots and perform statistical inferences like model assessments. Use cases are demonstrated using two real data sets contained in **metapack**.

1 Introduction

The U.S. Food and Drug Administration provides a clear definition of meta-analysis as *"the combining of evidence from relevant studies using appropriate statistical methods to allow inference(s) to be made to the population of interest"* (U.S. Food and Drug Administration et al., 2018). In fields like medicine, pharmacology, and epidemiology, meta-analysis has become popular for reconciling conflicting results or corroborating consistent ones in multiple studies (Chalmers et al., 2002; Borenstein et al., 2011; Hartung et al., 2011; Balduzzi et al., 2019). Findings produced from meta-analyses are often placed at the apex of the evidence hierarchy (U.S. Food and Drug Administration et al., 2018).

R already has a large supply of meta-analysis packages. **meta** (Schwarzer, 2007) and **rmeta** (Lumley, 2018) use the method of moments introduced in DerSimonian and Laird (1986). **metafor** (Viechtbauer, 2010) further contains moderator analyses and fits meta-regression (Berkey et al., 1995) through weighted least squares. On the other hand, **metaLik** (Guolo, 2012) takes a likelihood approach based on the second-order approximation of the modified likelihood ratio test statistic (Skovgaard et al., 1996). **metatest** (Huizenga et al., 2011) further includes hypothesis testing capabilities through the likelihood ratio test with Barlett correction, and **mvmeta** (Gasparrini et al., 2012) fits multivariate meta-analysis and meta-regression models via the method of maximum likelihood. There are packages for Bayesian meta-analytic inference as well. **bayesmeta** (Röver, 2020) assumes the normal-normal hierarchical random-effect model and allows the user to choose prior distributions with a great deal of flexibility, both informative and noninformative. **nmaINLA** (Guenhan et al., 2018) provides functionalities for network meta-analysis and meta-regression with integrated nested Laplace approximations (INLA) as an alternative to the Markov chain Monte Carlo (MCMC) algorithm. On the other hand, **bmeta** (Ding and Baio, 2016) delivers flexible meta-analytic modeling by interfacing with JAGS (Plummer, 2003). **MetaStan** (Guenhan, 2020) provides binomial-normal hierarchical models with weakly informative priors, building upon the probabilistic language Stan (Stan Development Team, 2020).

Despite its importance and the wide array of R packages available, meta-analysis is still regarded as a niche field that interests a narrow group of researchers and remains relatively low impact. We partially attribute this phenomenon to the fact that the R community has yet to come up with a user interface that unifies the theoretical distinctions between univariate and multivariate models, and between meta-analysis and network meta-analysis although the models have grown more complicated in the intervening years. Furthermore, a large class of simple Bayesian meta-analytic models is handled by probabilistic programming languages like Stan (Stan Development Team, 2020) or BUGS (Sturtz et al., 2005). Meanwhile, many complex models are not easily programmable in probabilistic languages, and are not readily available in R. Especially, in the context of variance-covariance matrix modeling in (network) meta-analysis, **metapack** is the first attempt in the R cosmos, to the best of our knowledge, to provide easy access to regression-modeling of the variances (of the treatment effects) as well as a wide array of options for modeling the response covariance matrices when the aggregate responses are multivariate with only a partially observed within-study sample covariance matrix.

metapack presented in this paper proposes a formula structure that flexibly represents the types of responses (univariate and multivariate) and the number of treatments (meta-analysis and network meta-analysis). The package also provides functions to assess model fits such as the deviance information criterion (DIC) and the logarithm of the pseudo-marginal likelihood (LPML), and to generate diagnostic plots. Some potential complications, theoretical and computational, in these model selection criteria may break the algorithm or erode the statistical inference when unaddressed (see Section Model comparison), which **metapack** takes care of by default—an advantage over model-agnostic programming languages.

The rest of this paper is organized as follows. Section Considered models briefly reviews the (network) meta-analysis model. Section Meta-analytic data for **metapack** describes the general form of a meta-analysis data set to establish a generic data structure for meta-analysis and network meta-analysis. Section Basic implementation of **metapack** explains how the data structure can be represented using R's extended formula and how **metapack**'s main function parses it, and lays down the various modeling options for meta-analysis and network meta-analysis. Section Performing inference further introduces the S3 methods available for performing statistical inferences and comparing models. Some computational considerations to accelerate the computation are detailed as well. Section Demonstration with real data provides demonstrations using the *cholesterol* data and *TNM* data included in **metapack**. Finally, Section Discussion concludes the paper by offering a cautionary remark for multivariate meta-analysis models regarding the number of observations required to perform valid inferences on the correlation matrix, and discussing future research and package development directions.

2 Considered models

In this section, we briefly review the models considered in **metapack**. There are largely two umbrella models: univariate or multivariate meta-analysis based on Yao et al. (2015), and univariate network meta-analysis based on Li et al. (2021). The various modeling options for each model introduced in this section *encompass* the ones in Yao et al. (2015) and Li et al. (2021). In what follows, the model description will deal with a general multivariate model including both meta-analysis and network meta-analysis, which is valid even when univariate response is assumed, i.e. $J = 1$. Occasionally, the univariate model description will be provided side by side to avoid confusion.

Assume K randomized controlled trials (RCTs) where the k-th trial includes T_k treatment arms. Meta-analysis refers to a special case where $T_k = 2$ for all $k = 1, \ldots, K$. We adopt the notational abuse of omitting the trial indicator in the treatment's subscript. For instance, for the sample size of the t-th treatment arm of the k-th trial, we write n_{kt}, not $n_{t_k k}$. Furthermore, for readability, note that boldface lowercase Latin letters are vectors, while uppercase Latin letters are matrices. Boldface Greek letters can either be vectors or matrices and will be defined contextually. Let $\boldsymbol{y}_{kt}$ be a J-dimensional aggregate response vector for the t-th treatment of the k-th trial. Similarly, let $\boldsymbol{x}_{ktj} \in \mathbf{R}^{p_j}$ be the treatment-within-trial level covariate corresponding to the j-th response, reflecting the fixed effects of the t-th treatment arm, and let $\boldsymbol{w}_{ktj} \in \mathbf{R}^{q_j}$ be the vector of covariates for the random effects. The model $\boldsymbol{y}_{kt} = \boldsymbol{X}_{kt}\boldsymbol{\beta} + \boldsymbol{W}_{kt}\gamma_k + \boldsymbol{\epsilon}_{kt}$ describes the aggregate data model, where $\boldsymbol{\epsilon}_{kt} = (\epsilon_{kt1}, \ldots, \epsilon_{ktJ})^\top$, $\boldsymbol{X}_{kt} = \oplus_{j=1}^{J} x_{ktj}^\top$ for which $\oplus$ indicates direct sum, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_J^\top)^\top$, $\boldsymbol{W}_{kt} = \oplus_{j=1}^{J} \boldsymbol{w}_{ktj}^\top$, and $\gamma_k = (\gamma_{k1}^\top, \gamma_{k2}^\top, \ldots, \gamma_{kJ}^\top)^\top$. The aggregate (network) meta-analysis model becomes

$$\begin{cases} \boldsymbol{y}_{kt} = \boldsymbol{X}_{kt}\boldsymbol{\beta} + \boldsymbol{W}_{kt}\gamma_k + \boldsymbol{\epsilon}_{kt} \text{ and } (n_{kt}-1)\boldsymbol{S}_{kt} \sim \mathcal{W}_{n_{kt}-1}(\Sigma_{kt}), & \text{if } J \geq 2 \text{ (multivariate)} \\ y_{kt} = \boldsymbol{x}_{kt}^\top\boldsymbol{\beta} + \boldsymbol{w}_{kt}^\top\gamma_k + \epsilon_{kt} \text{ and } (n_{kt}-1)s_{kt}^2/\sigma^2 \sim \chi^2_{n_{kt}-1}, & \text{if } J = 1 \text{ (univariate)} \end{cases}, \tag{1}$$

where $\boldsymbol{\epsilon}_{kt} \sim \mathcal{N}(\mathbf{0}, \Sigma_{kt}/n_{kt})$ or $\epsilon_{kt} \sim \mathcal{N}(0, \sigma_{kt}^2/n_{kt})$, $\boldsymbol{S}_{kt}$ is the sample covariance matrix, s_{kt}^2 is the sample variance, and $\Sigma_{kt} \in \mathcal{S}_{++}^J$ for which $\mathcal{S}_{++}^J$ is the space of $J \times J$ symmetric positive-definite matrices. In Equation (1), $\mathcal{W}_\nu(\Sigma)$ is the Wishart distribution with ν degrees of freedom and a $J \times J$ scale matrix Σ with density

$$p(X \mid \nu, \Sigma) = \frac{1}{2^{J\nu}|\Sigma|^{\nu/2}\Gamma_J(\nu/2)}|X|^{(\nu-J-1)/2}\exp\left(-\frac{1}{2}\mathrm{tr}(\Sigma^{-1}X)\right) I(X \in \mathcal{S}_{++}^J),$$

where Γ_J is the multivariate gamma function defined by $\Gamma_J(z) = \pi^{J(J-1)/4}\prod_{j=1}^{J}\Gamma[z + (1-j)/2]$. χ_ν^2 indicates the chi-squared distribution with ν degrees of freedom.

Stacking the random effects for all response endpoints, $\gamma_k = (\gamma_{k1}^\top, \ldots, \gamma_{kJ}^\top)^\top$. Since the random effects are assumed to follow a distribution in the location-scale family (either a multivariate t-distribution or a multivariate normal distribution), i.e. $\gamma_k \sim \mathcal{LS}(\gamma, \Omega)$, the fixed-effect coefficient

vector $\boldsymbol{\beta}$ absorbs the random effects' location parameter γ, forming $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \gamma^\top)^\top$. The corresponding design matrix $\boldsymbol{X}_{kt}$ is also expanded to include the random-effect design matrix $\boldsymbol{W}_{kt}$, written as $\boldsymbol{X}^*_{kt} = [\boldsymbol{X}_{kt}, \boldsymbol{W}_{kt}]$. With $\gamma_{k,o} = \gamma_k - \gamma$, the model now becomes

$$\begin{cases} \boldsymbol{y}_{kt} = \boldsymbol{X}^*_{kt}\boldsymbol{\theta} + \boldsymbol{W}_{kt}\gamma_{k,o} + \boldsymbol{\epsilon}_{kt}, & \text{if } J \geq 2 \text{ (multivariate)} \\ y_{kt} = \boldsymbol{x}^{*\top}_{kt}\boldsymbol{\theta} + \boldsymbol{w}^\top_{kt}\gamma_{k,o} + \epsilon_{kt} & \\ \quad \Rightarrow \boldsymbol{y}_k = \boldsymbol{X}^*_k\boldsymbol{\theta} + \boldsymbol{W}_k\gamma_{k,o} + \boldsymbol{\epsilon}_k, & \text{if } J = 1 \text{ (univariate)} \end{cases}, \tag{2}$$

where $\boldsymbol{y}_k = (y_{k,t_{k1}}, \ldots, y_{k,t_{kT_k}})^\top$, $\boldsymbol{X}^*_k = ((\boldsymbol{x}^*_{k,t_{k1}})^\top, \ldots, (\boldsymbol{x}^*_{k,t_{kT_k}})^\top)^\top$, $\boldsymbol{W}_k = (\boldsymbol{w}^\top_{k,t_{k1}}, \ldots, \boldsymbol{w}^\top_{k,t_{kT_k}})^\top$, and $\boldsymbol{\epsilon}_k = (\epsilon_{k,t_{k1}}, \ldots, \epsilon_{k,t_{kT_k}})^\top$. We briefly restore the correct subscripts (i.e. t_{kl} for $l = 1, \ldots, T_k$) to demonstrate that $\boldsymbol{y}_k$, $\boldsymbol{X}^*_k$, and $\boldsymbol{W}_k$ may be different lengths and dimensions for different k's. Here, $\{t_{k1}, \ldots, t_{kT_k}\}$ denotes the set of treatments compared in the k-th trial. The random effects are modeled differently in meta-analysis than in network meta-analysis. A major reason for this divergence is that the variables explaining the treatment effects are not easily found in the presence of varying numbers of treatments in different trials. The differences are further detailed in Section Meta-analysis models and Section Network meta-analysis models.

3 Meta-analytic data for metapack

To streamline configuring models in R formula, it is important to understand the data structure for **metapack**. Table An example of arm-level meta-analytic data. **Trial** is equivalent to Study ID. A meta-analysis has two treatments in each trial for all trials, whereas a network meta-analysis can have trials with different numbers of treatments across trials. This distinction determines the number of rows for each trial (i.e. strictly two rows per trial in meta-analyses and a differing number of rows per trial for network meta-analyses). `Outcome`, `SD`, `DeisngM1`, and `DesignM2` can each be a vector, in which case the row vector representation indicates distributing across columns. For example, if `Outcome` consists of two endpoints, (Y_1, Y_2), then each $\boldsymbol{y}^\top_{kt}$ should enter a row as two columns, `y1` and `y2`. represents a typical arm-level data set for (network) meta-analysis, where each row represents a trial arm.

Outcome ($\boldsymbol{y}_{kt}$)	**SD** (s_{kt})	**DesignM1** ($\boldsymbol{x}_{kt}$)	**DesignM2** ($\boldsymbol{w}_{kt}$)	**Trial** (k)	**Treat** (t)	**n**
$\boldsymbol{y}^\top_{14}$	s_{14}	$\boldsymbol{x}^\top_{14}$	$\boldsymbol{w}^\top_{14}$	1	4	1000
$\boldsymbol{y}^\top_{11}$	s_{11}	$\boldsymbol{x}^\top_{11}$	$\boldsymbol{w}^\top_{11}$	1	1	545
$\boldsymbol{y}^\top_{21}$	s_{21}	$\boldsymbol{x}^\top_{21}$	$\boldsymbol{w}^\top_{21}$	2	1	1200
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1: An example of arm-level meta-analytic data. **Trial** is equivalent to Study ID. A meta-analysis has two treatments in each trial for all trials, whereas a network meta-analysis can have trials with different numbers of treatments across trials. This distinction determines the number of rows for each trial (i.e. strictly two rows per trial in meta-analyses and a differing number of rows per trial for network meta-analyses). `Outcome`, `SD`, `DeisngM1`, and `DesignM2` can each be a vector, in which case the row vector representation indicates distributing across columns. For example, if `Outcome` consists of two endpoints, (Y_1, Y_2), then each $\boldsymbol{y}^\top_{kt}$ should enter a row as two columns, `y1` and `y2`.

`Outcome` is the response (or responses for multivariate cases), `SD` is the standard deviation(s) of the response(s), `DesignM1` and `DesignM2` are design matrices, and `n` is the arm sample size. The pair of trial and treatment indicators is unique to a row. The first design matrix, `DesignM1`, contains the covariates for fixed effects and will be written as $\boldsymbol{X}$ henceforth. The second design matrix, `DesignM2` or $\boldsymbol{W}$, represents different things depending on the model, which will be explained in Section Meta-analysis models and Section Network meta-analysis models. It should be noted that there can always be two design matrices, whose configuration will be illustrated in Section Basic implementation of **metapack**.

A meta-analytic data set is characterized as folows: (1) univariate or multivariate and (2) meta-analysis or network meta-analysis. Here, meta-analysis refers to when trials have specifically two treatments (i.e. $t = 1, 2$ for all k) and all treatments are compared head to head. On the other hand, network meta-analysis includes more than two total treatments, where each trial can have a different set of treatments, allowing indirect comparison between treatments that are not compared head to head. The data structure is unchanged for network meta-analysis except that `Treat` can have more than two unique values. The first category (univariate vs. multivariate) is determined by the number of response endpoints, and the second category (meta- vs. network meta-analysis) by the number of

treatments. All other modeling choices fall into prior specification.

4 Basic implementation of metapack

`bmeta_analyze` is the main function in **metapack**, whose first argument is an R formula. `bmeta_analyze` internally parses a formula to identify a model and ultimately calls a *worker function*. An extension of R's formula class, **Formula** (Zeileis and Croissant, 2010), accommodates multiple responses and parts, lending itself well into meta-analysis modeling. Once a model is fully identified, the MCMC algorithm is executed in C++, thanks to **Rcpp** (Eddelbuettel and Balamuta, 2017) and **RcppArmadillo** (Eddelbuettel and Sanderson, 2014).

Name	Functionality	Description
`bmeta_analyze`	Estimation	Fits (network) meta-analysis models
`hpd`	Inference	Computes highest posterior density (HPD) intervals of model parameters
`model_comp`	Inference	Computes model comparison measures (DIC or LPML)
`print`		Displays a summary of the output
`summary`		Displays a summary of the output
`plot`		Plots trace plots, density plots, or surface-under-the-cumulative-ranking-curve (SUCRA) plots
`fitted`		Computes posterior means, standard deviations, and HPD intervals
`coef`		Computes posterior means of fixed-effect coefficients
`cholesterol`	Data set	Cholesterol data for multivariate meta-analysis
`TNM`	Data set	Triglyceride data for network meta-analysis

Table 2: A list of available functions and data sets in **metapack**.

Using `Formula`

The three characterizations of a meta-analytic data set must be encoded in the formula. Requiring the formula to have two left-hand sides (LHS) and two or three right-hand sides (RHS)[1] is enough to communicate the characterizations for a wide class of meta-analysis models. We invite other R package developers to adopt the following representation for meta-analytic models, the general form of which is given by

```
y1 + y2 | sd1 + sd2 ~ x1 + x2 + x3 + ns(n) | w1 + w2 + w3 | treat + trial (+ groups)
```

Each *part* in LHS or RHS is an R expression where variables (or functions of variables) are chained with a plus sign (+)—e.g. `x1 + x2`. The tilde (~) separates all LHSs from all RHSs, each further separated into parts by vertical bars (|). The meaning of each part is syntactically determined by its location within the formula, like an English sentence. Therefore, every part must come in the exact order as prescribed for `bmeta_analyze` to correctly identify the model.

- The first LHS ('`y1 + y2`'), the responses, is required of all models. Depending on the number of variables given in the first LHS, `bmeta_analyze` will determine whether the model is multivariate or univariate. For example, a first LHS with only `y` would flag the model as univariate.
- The second LHS ('`sd1 + sd2`') supplies the standard deviations of the endpoints required of an aggregate-data meta-analysis. The function call will break if this part is missing.
- The first RHS ('`x1 + x2 + x3 + ns(n)`') defines the fixed-effect covariates. For aggregate-data models, the arm sample sizes must be passed as an argument to `ns()`. In the example code, `n` is the variable containing the arm sample sizes.
- The second RHS ('`w1 + w2 + w3`') defines either the random-effect covariates ($\boldsymbol{w}_{kt}^\top \boldsymbol{\gamma}$) or the variance-related covariates ($\log \tau_{kt} = \boldsymbol{w}_{kt}^\top \boldsymbol{\phi}$)—see Sections Meta-analysis models or Network meta-analysis models for details. This part is optional. If omitted, `bmeta_analyze` will assume $\boldsymbol{w}_{kt} = \mathbf{1}$ where $\mathbf{1}$ is a vector of ones.
- The third RHS ('`treat + trial`' or '`treat + trial + groups`') corresponds to the treatment and trial indicators, and optionally the grouping variable if it exists. Variables here must be provided in the exact order shown in the example.

The dimension of the response(s) is explicit in the formula, which determines the first characterization. The treatments are coerced to a factor—if not already one—whose number of levels is extracted (i.e. '`nlevels(treat)`') to resolve the second characterization, meta-analysis versus network meta-analysis.

[1] Semantically, the LHS should refer to the entire left of tilde—same for RHS—but in R idioms, when a side is counted or pluralized (e.g. LHSs, RHSs, or sides), it refers to a part or parts in the corresponding side.

Function arguments

Aside from the first two arguments, `formula` and `data`, there are four other optional arguments that must be provided as R's `list` class: `prior`, `mcmc`, `control`, and `init`. All hyperparameters for the prior distributions should be included in `prior`—see Section Considered models for hyperparameters. `mcmc` only regards the numbers of MCMC iterations: `ndiscard` for the number of burn-in iterations, `nskip` for thinning, and `nkeep` for the posterior sample size. `control` configures the Metropolis-Hastings algorithm. `*_stepsize` with the asterisk replaced with one of the parameter names indicates the step size for determining the sample evaluation points in the localized Metropolis algorithm. Lastly, initial values for the parameters can be provided in `init` in case a user has *a priori* known high-quality starting points.

Meta-analysis models

For meta-analysis models, **metapack** acknowledges the possible existence of the first-line and second-line treatments trials. More generally, the trials may be grouped by a factor believed to generate disparate random effects. Although an arbitrary number of groups can exist in theory, we restrict our attention to two groups. Denoting the binary group indicators by $u_{kt} \in \{0, 1\}$ yields

$$y_{ktj} = \boldsymbol{x}_{ktj}^\top \boldsymbol{\beta} + (1 - u_{kt}) \boldsymbol{w}_{ktj}^\top \gamma_{kj}^0 + u_{kt} \boldsymbol{w}_{ktj}^\top \gamma_{kj}^1 + \epsilon_{ktj}. \tag{3}$$

The random effects are modeled as $\gamma_{kj}^l \overset{\text{ind}}{\sim} \mathcal{N}(\gamma_j^{l*}, \Omega_j^l)$ and $(\Omega_j^l)^{-1} \sim \mathcal{W}_{d_{0j}}(\Omega_{0j})$. Stacking the vectors, $\gamma_k^l = ((\gamma_{k1}^l)^\top, \ldots, (\gamma_{kJ}^l)^\top)^\top \sim \mathcal{N}(\gamma^{l*}, \Omega^l)$, where $\gamma^{l*} = ((\gamma_1^{l*})^\top, \ldots, (\gamma_J^{l*})^\top)^\top$, $\Omega_j = \Omega_j^0 \oplus \Omega_j^1$, and $\Omega = \oplus_{j=1}^J \Omega_j$ for $l \in \{0, 1\}$. Adopting the *noncentered parameterization* (Bernardo et al., 2003), define $\gamma_{k,o}^l = \gamma_k^l - \gamma^{l*}$. Denoting $\boldsymbol{W}_{kt}^* = [(1 - u_{kt})\boldsymbol{W}_{kt}, u_{kt}\boldsymbol{W}_{kt}]$, $\boldsymbol{X}_{kt}^* = [\boldsymbol{X}_{kt}, \boldsymbol{W}_{kt}^*]$, $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \gamma^{0*\top}, \gamma^{1*\top})^\top$, and $\gamma_{k,o} = ((\gamma_{k,o}^0)^\top, (\gamma_{k,o}^1)^\top)^\top$, the model is written as follows:

$$\boldsymbol{y}_{kt} = \boldsymbol{X}_{kt}^* \boldsymbol{\theta} + \boldsymbol{W}_{kt}^* \gamma_{k,o} + \boldsymbol{\epsilon}_{kt}. \tag{4}$$

If there is no distinction between the first-line and second-line therapies, then setting $u_{kt} = 0$ for all (k, t) reduces the model back to Equation (1). Finally, we assume $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, c_0 \boldsymbol{I})$ where $\boldsymbol{I}$ is an identity matrix.

A (boilerplate) template for this class of models is as follows:

```
f <- "y1 + y2 | sd1 + sd2 ~ x1 + x2 | w1 + w2 | treat + trial + groups"
fit <- bmeta_analyze(formula(f), data = df,
        prior = list(c0 = [real], dj0 = [real], Omega0 = [matrix],
                     a0 = [real], b0 = [real],
                     d0 = [real], nu0 = [real], Sigma0 = [matrix]),
        control = list(sample_Rho = [logical], Rho_stepsize = [real],
                       R_stepsize = [real], delta_stepsize = [real], model = [string]))
```

We use '`real`' and '`string`' as aliases for '`double`' and '`character`' in R. Every bracketed expression should be replaced with an instance of the enclosed class. The hyperparameters in '`prior`' and step sizes in '`control`' will be clarified in the following modeling options. Note that all parameters with a step size are sampled through the Metropolis-Hastings algorithm.

Modeling options for Σ_{kt} The covariance matrix between the response endpoints (Σ_{kt}) can be modeled depending on (1) the amount of data available; and (2) what assumptions the practitioner is willing to make. The diagonal elements of Σ_{kt} are always identifiable whereas the off-diagonal elements require additional modeling assumptions. **metapack** presently offers five options, specifiable through `model` in the `control` argument. For $\mathcal{M}_2$–$\mathcal{M}_5$, the unobserved sample correlation matrices are sampled from their conditional distributions $(R_{kt} \mid V_{kt}, \Sigma_{kt})$ given by

$$p(R_{kt} \mid V_{kt}, \Sigma_{kt}) \propto |R_{kt}|^{(n_{kt}-J-2)/2} \exp\left\{-\frac{(n_{kt}-1)}{2} \operatorname{tr}\left(V_{kt}^{\frac{1}{2}} \Sigma_{kt}^{-1} V_{kt}^{\frac{1}{2}} R_{kt}\right)\right\}. \tag{5}$$

- ($\mathcal{M}_1$: `model="NoRecovery"`) The simplest and easiest way to model the covariance matrices is to relinquish correlation recovery, in which case Equation (5) is ignored. We assume $\Sigma_{kt} = \operatorname{diag}(\sigma_{kt,11}^2, \ldots, \sigma_{kt,JJ}^2)$ with $\sigma_{kt,jj}^2 \sim \mathcal{IG}(a_0, b_0)$ for $a_0, b_0 > 0$, where $\mathcal{IG}(a, b)$ denotes the inverse-gamma distribution whose density function is proportional to $x^{-(a+1)} \exp(-b/x)$ for $x > 0$. For univariate meta-analyses, this is the only valid option since there are no off-diagonal entries.

- ($\mathcal{M}_2$: `model="EquiCovariance"`) We can assume the covariance matrix is for all pairs of treatments and trials. That is, $\Sigma_{kt} = \Sigma$ for every combination of (k, t) for $t = 1, \ldots, T$ and $k = 1, \ldots, K$. A Wishart prior distribution is assumed for Σ^{-1}, i.e. $\Sigma^{-1} \sim \mathcal{W}_{s_0}(\Sigma_0)$ for $s_0 > J - 1$ and $\Sigma_0 \in \mathcal{S}^J_{++}$.
- ($\mathcal{M}_3$: `model="EquiWithinTreat"`) If the equal covariance is too strong an assumption, we can allow the covariance matrices to be equivalent within a treatment. That is, $\Sigma_{kt} = \Sigma_t$ for $t = 1, \ldots, T$. Similar to $\mathcal{M}_2$, a Wishart prior distribution is assumed for Σ_t^{-1}, i.e. $\Sigma_t^{-1} \sim \mathcal{W}_{s_0}(\Sigma_0)$.
- ($\mathcal{M}_4$: `model="EquiCorrelation"`) To achieve the best of both worlds—variances and correlations—we can let the variances enjoy maximum freedom but attempt to recover the correlations by restricting them to be identical across treatments and trials for identifiability. Performing the decomposition,

  $$\Sigma_{kt} = \boldsymbol{\delta}_{kt} \boldsymbol{\rho} \boldsymbol{\delta}_{kt},$$

  where $\boldsymbol{\delta}_{kt} = \text{diag}(\Sigma^{1/2}_{kt,11}, \ldots, \Sigma^{1/2}_{kt,JJ})$, and $\boldsymbol{\rho}$ is the correlation matrix, the elements in $\boldsymbol{\delta}_{kt}$ and $\boldsymbol{\rho}$ are sampled through the Metropolis-Hastings algorithm.
- ($\mathcal{M}_5$: `model="Hierarchical"`) The hierarchical prior for Σ_{kt} is given by $(\Sigma_{kt}^{-1} \mid \Sigma) \sim \mathcal{W}_{\nu_0}((\nu_0 - J - 1)^{-1}\Sigma^{-1})$ and $\Sigma \sim \mathcal{W}_{d_0}(\Sigma_0)$. By allowing the Σ_{kt}'s to differ but having them share information across treatments and trials via Σ, this assumption aims to control the between-treatment and between-trial variations simultaneously. Since the amount of information shared between treatments and trials is controlled by ν_0 ($> J - 1$), it is advised to try multiple values for ν_0 and perform a model assessment through the deviance information criterion (DIC) or the logarithm of the pseudo-marginal likelihood (LPML). Σ is further decomposed for Metropolis-Hastings algorithm as $\Sigma = \Delta\boldsymbol{\rho}\Delta$ where $\Delta = \text{diag}(\delta_1, \ldots, \delta_J)$ is the diagonal matrix of standard deviations, and $\boldsymbol{\rho}$ is a correlation matrix with unit diagonal elements.

Network meta-analysis models

For univariate network meta-analysis, the design matrix for random effects is restricted to be the selection matrix $E_k^\top = (e_{t_{k1}}, e_{t_{k2}}, \ldots, e_{t_{kT_k}})^\top$, where $e_{t_{kl}} = (0, \ldots, 1, \ldots, 0)^\top$, $l = 1, \ldots, T_k$, with the t_{kl}th element set to 1 and 0 otherwise, and T_k is the number of treatments included in the k-th trial. Furthermore, we redefine $\gamma_{k,o}$ to be $\gamma_{k,o} := E_k^\top \gamma_k$, a vector of T_k-dimensional scaled random effects. The random effects $\gamma_k \sim t_T(\boldsymbol{\gamma}, \boldsymbol{\rho}, \nu)$, where $t_T(\boldsymbol{\mu}, \Sigma, \nu)$ denotes a multivariate t-distribution with ν degrees of freedom, a location parameter vector $\boldsymbol{\mu}$, and a scale matrix Σ. T indicates the number of distinct treatments in all trials. The random effects γ_k are scaled since $\boldsymbol{\rho}$ is a correlation matrix with unit diagonal entries, and the variance components can be modeled as a multiplicative term. That is, with $\boldsymbol{W}_k(\boldsymbol{\phi}) = \text{diag}(\exp(\boldsymbol{w}_{kt_{k1}}^\top \boldsymbol{\phi}), \ldots, \exp(\boldsymbol{w}_{kt_{kT_k}}^\top \boldsymbol{\phi}))$, the model is recast as

$$\boldsymbol{y}_k = \boldsymbol{X}_k^* \boldsymbol{\theta} + \boldsymbol{W}_k(\boldsymbol{\phi})\gamma_{k,o} + \boldsymbol{\epsilon}_k,$$

where $\boldsymbol{X}_k^* = (\boldsymbol{X}_k, E_k^\top)$, $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top)^\top$, $\boldsymbol{\epsilon}_k \sim \mathcal{N}_{T_k}(\boldsymbol{0}, \Sigma_k)$, and $\Sigma_k = \text{diag}\left(\frac{\sigma^2_{kt_{k1}}}{n_{kt_{k1}}}, \ldots, \frac{\sigma^2_{kt_{kT_k}}}{n_{kt_{kT_k}}}\right)$. This allows $\exp(\boldsymbol{w}_{kt}^\top \boldsymbol{\phi})$ to be the standard deviation of γ_{kt}. Since the multivariate t-random effects are not analytically marginalizable, we represent it as a scale mixture of normals as

$$(\gamma_{k,o} \mid \lambda_k) \overset{\text{ind}}{\sim} \mathcal{N}_{T_k}\left(\boldsymbol{0}, \lambda_k^{-1}(E_k^\top \boldsymbol{\rho} E_k)\right), \qquad \lambda_k \overset{\text{iid}}{\sim} \mathcal{G}a\left(\frac{\nu}{2}, \frac{\nu}{2}\right), \tag{6}$$

where $\mathcal{G}a(a, b)$ indicates the gamma distribution with mean a/b. Finally, $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, c_{01}\boldsymbol{I})$ and $\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{0}, c_{02}\boldsymbol{I})$.

A (boilerplate) template for this class of models is as follows:

```
f <- "y | sd ~ x1 + x2 | w1 + w2 | treat + trial"
fit <- bmeta_analyze(formula(f), data = df,
          prior = list(c01 = [real], c02 = [real], df = [real],
            a4 = [real], b4 = [real], a5 = [real], b5 = [real]),
          control = list(sample_df = [logical], sample_Rho = [logical],
            Rho_stepsize = [real], phi_stepsize = [real], lambda_stepsize = [real]))
```

Every bracketed expression should be replaced with an instance of the enclosed class. The hyperparameters in '`prior`' and step sizes in '`control`' will be clarified in the following modeling options. Note that all parameters with a step size are sampled through the Metropolis-Hastings algorithm.

Modeling options The appeal of considering heavy-tailed random effects and modeling the variance to be a deterministic linear function of a covariate is that both extend but still cover the common cases (normal random effects and no variance modeling) as either the limiting or a special case. Unlike meta-analysis, there is no single argument ('`model`') determining the modeling option. A model is rather specified by a combination of arguments.

- ($\mathcal{M}_1$ - No variance modeling) Sometimes, no covariate information is available for modeling the variances, where $\boldsymbol{w}_{kt_{kt}}$ reduces to one—$\boldsymbol{W}_k(\boldsymbol{\phi}) = \text{diag}(e^{\phi}, \ldots, e^{\phi})$. The marginal variance of y_{kt} becomes $\text{Var}(y_{kt}) = \sigma_{kt}^2/n_{kt} + e^{2\phi}$. This can be achieved by simply omitting the *second RHS*, thereby making the trial and treatment configuration the second RHS. For example, '`y | sd ~x1 + x2 | treat + trial`'.
- ($\mathcal{M}_2$ - Normal random effects) In the limiting case where $\nu \to \infty$, $\gamma_{k,o} \sim \mathcal{N}_{T_k}(0, E_k^\top \boldsymbol{\rho} E_k)$. `bmeta_analyze` treats ν ('`df`') as part of the prior specification. Thus, '`df=Inf`' in the '`prior`' argument corresponds to opting for normal random effects.
- ($\mathcal{M}_3$ - Random degrees of freedom) The degrees of freedom of the multivariate t-distribution for the random effects are treated as unknown. In this case, a hierarchical prior is considered for the degrees of freedom. That is, $(\nu \mid \nu_a, \nu_b) \sim \mathcal{G}a(\nu_a, \nu_a/\nu_b)$, $\nu_a \sim \mathcal{G}a(a_4, b_4)$, and $\nu_b \sim \mathcal{IG}(a_5, b_5)$. Since sampling ν regards the MCMC algorithm, a logical variable '`sample_df`' is placed in the '`control`' argument. If '`sample_df=TRUE`', the value for '`df`' in '`prior`' will be used as the initial value, and the related hyperparameters ('`a4`', '`b4`', '`a5`', and '`b5`') can be set in '`prior`'. For obvious reasons, '`df`' cannot be assigned '`Inf`' if '`sample_df=TRUE`'.

5 Performing inference

The object (`fit`) returned from `bmeta_analyze` remembers the function arguments, encapsulates the model specification, and contains the posterior sample from the MCMC algorithm. Therefore, this object alone can be passed to other methods to perform subsequent inferences. These methods include `fitted`, `hpd`, `coef`, `model_comp`, and `sucra`.

The posterior means, standard deviations, and the HPD intervals are computed via the `fitted()` function. The `fitted()` function has two optional arguments: '`level`' and '`HPD`'. '`level`' determines the credibility level of the interval estimation. '`HPD`', a logical parameter, decides whether a highest posterior density or equal-tailed credible interval will be produced. It is also possible to obtain the posterior interval estimates only using `hpd()`.

```
R> est <- fitted(fit, level=0.99, HPD = TRUE)
R> hpd <- hpd(fit, level = 0.95, HPD = TRUE)
```

'`coef(fit)`' allows users to extract the posterior mean of the fixed-effect coefficients.

Model comparison It is crucial to determine a suitable model to base the statistical inference on. Section Meta-analysis models introduces five models for Σ_{kt}, and Section Network meta-analysis models contains infinitely many models since the degrees of freedom ν for the random effects can assume any number on the positive real line including infinity. Such a circumstance calls for a principled way of comparing models as well as evaluating goodness of fit.

The deviance information criterion (Spiegelhalter et al., 2002), or DIC, is defined as follows:

$$\text{DIC} = \text{Dev}(\bar{\boldsymbol{\eta}}) + 2p_D,$$

where $\boldsymbol{\eta}$ indicates all model parameters for which $\bar{\boldsymbol{\eta}} = \mathbb{E}[\boldsymbol{\eta} \mid D_{\text{obs}}]$. $\text{Dev}(\boldsymbol{\eta})$ is the deviance function given by $\text{Dev}(\boldsymbol{\eta}) = -2 \log L_{o\mathbf{y}}(\boldsymbol{\eta} \mid D_{\text{obs}})$, for which $L_{o\mathbf{y}}$ is the observed-data likelihood associated with $\boldsymbol{y}$, and p_D is defined as $p_D = \overline{\text{Dev}(\boldsymbol{\eta})} - \text{Dev}(\bar{\boldsymbol{\eta}})$ where $\overline{\text{Dev}(\boldsymbol{\eta})} = \mathbb{E}[\text{Dev}(\boldsymbol{\eta}) \mid D_{\text{obs}}]$.

```
R> dic <- model_comp(fit, type = "dic", verbose = TRUE, ncores = 3)
```

Another Bayesian model selection criterion is the logarithm of the pseudo-marginal likelihood (LPML), defined as the summed logarithm of the conditional predictive ordinates (CPO). The CPO has a "leave-one-out" predictive interpretation, which in meta-analyses is often overlooked—a significant oversight that will undermine model comparison. The aggregate nature of meta-analysis data calls for a redefinition of what is "left out" and what should be the base unit for prediction. With trials as the base unit, the CPO for the k-th trial is

$$\text{CPO}_k = \int L(\boldsymbol{\eta} \mid D_{o\mathbf{y}}) p(\boldsymbol{\eta} \mid D_{o\mathbf{y}}^{-k}, D_{o\mathbf{s}}) \, \mathrm{d}\boldsymbol{\eta},$$

where $D_{o\mathbf{y}}^{-k}$ is $D_{o\mathbf{y}}$ with the k-th trial removed, and $p(\boldsymbol{\eta} \mid D_{o\mathbf{y}}^{-k}, D_{o\mathbf{s}})$ is the posterior distribution based on the data without the k-th trial. Then, LPML $= \frac{1}{K}\sum_{k=1}^{K} \log(\text{CPO}_k)$.

```
R> lpml <- model_comp(fit, type = "lpml", verbose = TRUE, ncores = 3)
```

Treatments included in only one trial violate the predictive interpretation of the CPO. The corresponding trials thus should be removed from LPML calculation. `model_comp` returns the logarithm of the CPO of every trial but corrects the LPML should such treatments exist. For those occasions, a naive sum of the logarithm of the CPO_k's will not equal the corrected LPML in the returned object.

The DIC and LPML for the meta-analysis models are relatively straightforward since the observed-data likelihood is a multivariate normal density. However, network meta-analysis models require some technical considerations in evaluating the following observed-data likelihood that involves an analytically intractable integral when t-random effects are used: slow computation speed and numerical overflow. Observe the following observed likelihood function for the network meta-analysis model:

$$L_{o\mathbf{y}}(\boldsymbol{\eta} \mid D_{\text{obs}}) = \prod_{k=1}^{K} \int_0^{\infty} (2\pi)^{-\frac{T_k}{2}} g(\lambda_k) \left| \lambda_k^{-1}(\boldsymbol{W}_k(\boldsymbol{\phi}) E_k^{\top} \boldsymbol{\rho} E_k \boldsymbol{W}_k(\boldsymbol{\phi})) + \Sigma_k \right|^{-\frac{1}{2}} \times \exp\left\{ -\frac{(\boldsymbol{y}_k - \boldsymbol{X}_k^{*}\boldsymbol{\theta})^{\top} \left[\lambda_k^{-1} \boldsymbol{W}_k(\boldsymbol{\phi}) E_k^{\top} \boldsymbol{\rho} E_k \boldsymbol{W}_k(\boldsymbol{\phi}) + \Sigma_k \right]^{-1} (\boldsymbol{y}_k - \boldsymbol{X}_k^{*}\boldsymbol{\theta})}{2} \right\} \mathrm{d}\lambda_k,$$

where $g(\lambda_k)$ is the gamma density with shape and rate parameters $\nu/2$. The integral is evaluated via double exponential (DE) quadrature, or equivalently tanh-sinh quadrature (Takahasi and Mori, 1974; Bailey et al., 2005), available in the **math** header of the **BH** package (Eddelbuettel et al., 2019). DE quadrature is robust to singularities, terminates fast, and provides high precision (Bailey et al., 2005). We address the slow speed through *"shared memory multiprocessing programming"* via **OpenMP** (OpenMP Architecture Review Board, 2018). **OpenMP** is a widely used application programming interface (API) for portable and scalable parallel processing in C, C++, and Fortran across many operating systems. As long as R is configured for **OpenMP**, **metapack** will deploy parallelism. Unless the argument `ncores` is specified otherwise, `model_comp` will use two CPU cores for parallel computing by default.

To prevent overflow, we take the following steps:

- Let $h(\lambda_k)$ denote the integrand. Then, compute $\widehat{\lambda}_k = \arg\max_{\lambda_k} \log h(\lambda_k)$.
- Redefine the integral as

$$\exp\left\{ \log h(\widehat{\lambda}_k) + \log \int_0^{\infty} \exp\left[\log h(\lambda_k) - \log h(\widehat{\lambda}_k) \right] \mathrm{d}\lambda_k \right\}.$$

Although the exponential shifting scheme does not warrant preventing every occurrence of numerical overflow, we have observed stable evaluations of the integral for over several thousand batches of simulations.

Model diagnostics and visualization

It is important to diagnose whether the results are consistent with the assumptions and visualize the findings. **metapack** provides methods for these: `plot` and `sucra`. The `plot` method is available for both meta-analysis and network meta-analysis whereas `sucra` is exclusively for network meta-analysis.

The `plot` method will take the `fit` object and generate the density plots and trace plots of $\boldsymbol{\theta}$. To see the plots for other parameters, run the following commands using **coda** (Plummer et al., 2006):

```
R> library("coda")
R> posterior <- as.mcmc(data.frame(gammaR = fit$mcmc.draws$gamR,
+    sig2 = fit$mcmc.draws$sig2))
R> plot(posterior)
```

Similarly, **boa** (Smith, 2007) can be used for output analysis. The posterior sample of $\boldsymbol{\rho}$ comes in three-dimensional arrays, which requires suitable indexing to generate trace plots for the off-diagonal lower-triangular elements. To generate such trace plots, run the following command:

```
R> idx <- lower.tri(fit$mcmc.draws$Rho[,,1])
R> n_idx <- ncol(idx) * (ncol(idx) - 1) / 2
R> posterior <- as.mcmc(data.frame(
```

```
+                 rho = t(vapply(1:fit$mcmc$nkeep, function(ikeep) {
+                   rho_i <- fit$mcmc.draws$Rho[,,ikeep]
+                   rho_i[idx]
+                 }, FUN.VALUE=numeric(n_idx)))))
R> plot(posterior)
```

Treatment comparisons The *surface under the cumulative ranking (SUCRA)* curve is useful when the ranking of the treatments in a network meta-analysis is of interest. Based on the posterior sample, $\mathbb{P}(t,r)$ denotes the probability that the treatment t is ranked $1 \leq r \leq T$ for $t = 1,\ldots,T$. Let $\mathbf{P}$ be the $T \times T$ discrete rank (row-stochastic) probability matrix whose (t,r)th element is $\mathbb{P}(t,r)$. The cumulative probability is then computed through $F(t,x) = \sum_{r=1}^{x} \mathbb{P}(t,r)$, where $F(t,x)$ is the probability that the t-th treatment is ranked x or better. Since $F(t,T) = 1$ for every t, the surface under the cumulative ranking distribution for the t-th distribution is given by

$$\text{SUCRA}(t) = \frac{1}{T-1} \sum_{x=1}^{T-1} F(t,x).$$

`sucra` will take the `fit` object and return the SUCRA and the discrete rank probability matrix $\mathbf{P}$.

```
R> s <- sucra(fit)
R> s$SUCRA
R> s$rankprob
```

If only plotting is needed, the storage of the `sucra` object can be bypassed via running `plot(sucra(fit))`. Note that SUCRA s has a bijection with the mean rank r of a treatment, $r = 1 + (1-s)(T-1)$, where T is the number of treatments.

6 Demonstration with real data

Meta-analysis

metapack includes a data set, `cholesterol`, which consists of 26 double-blind, randomized, active, or placebo-controlled clinical trials on patients with primary hypercholesterolemia sponsored by Merck & Co., Inc., Kenilworth, NJ, USA (Yao et al., 2015). The data set can be loaded by running `data("cholesterol")`. The `cholesterol` data set has three endpoints: low density lipoprotein cholesterol (`pldlc`), high density lipoprotein cholesterol (`phdlc`), and triglycerides (`ptg`). The percent change from the baseline in the endpoints, variables prefixed by `p-`, are the aggregate responses, followed by the corresponding standard deviations prefixed by `sd-`. Variable documentation is also available in `help("cholesterol")`.

```
R> set.seed(2797542)
R> f_1 <- 'pldlc + phdlc + ptg | sdldl + sdhdl + sdtg ~
+    0 + bldlc + bhdlc + btg + age + durat + white + male + dm + ns(n) | treat |
+    treat + trial + onstat'
R> fit_ma <- bmeta_analyze(formula(f_1), data = cholesterol,
+   prior = list(model="NoRecovery"),
+   mcmc = list(ndiscard = 1000, nkeep = 1000),
+   control=list(scale_x = TRUE, verbose=TRUE))
```

Variable	Description
`study`	study identifier
`trial`	trial identifier
`treat`	treatment indicator for Statin or Statin+Ezetimibe
`n`	the number of participants in the study arms
`pldlc`	aggregate percentage change in LDL-C
`phdlc`	aggregate percentage change from baseline in HDL-C
`ptg`	aggregate percentage change from baseline in triglycerides (TG)
`sdldl`	sample standard deviation of percentage change in LDL-C
`sdhdl`	sample standard deviation of percentage change in HDL-C
`sdtg`	sample standard deviation of percentage change in triglycerides (TG)
`onstat`	whether the participants were on Statin prior to the trial
`bldlc`	baseline LDL-C
`bhdlc`	baseline HDL-C
`btg`	baseline triglycerides (TG)
`age`	age in years
`white`	the proportion of white participants
`male`	the proportion of male participants
`dm`	the proportion of participants with diabetes mellitus
`durat`	duration in weeks

Table 3: Variables included in the `cholesterol` data set.

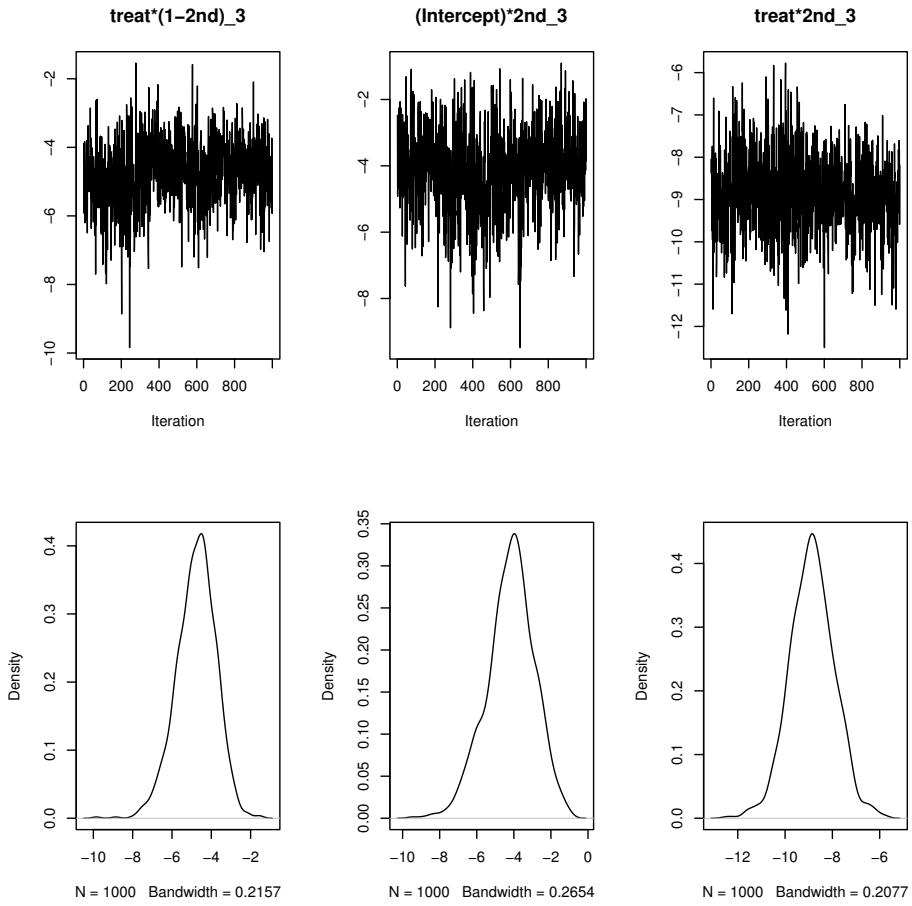


Figure 1: The trace plots (three top panels) and density plots (three bottom panels) of the treatment effects from the meta-analysis model generated from `plot(fit_ma)`—plots have been omitted for brevity. The trace plots show good mixing and convergence of MCMC chains and the density plots indicate that the marginal posterior distribution for each treatment effect is roughly symmetric. The MCMC samples of the regression coefficients will be automatically assigned row names according to the formula provided by the user. '`treat*(1-2nd)_3`' indicates the treatment effect within the first-line patients (i.e. '`onstat=0`') with respect to the third response variable, '`ptg`'. Likewise, '`(Intercept)*2nd_3`' is the baseline effect within the second-line patients (i.e. '`onstat=1`') with respect to '`ptg`'.

summary can be used to summarize the posterior sample from the 'fit' object. summary will name the variables accordingly in the output, suffixed by the index j in the corresponding outcome variable y_{ktj}. For example, 'bldlc_1' in the output below corresponds to the base LDL-C's coefficient associated with the first endpoint ('pldlc'). The summary table consists of the posterior mean, posterior standard deviation, the HPD lower bound, and the HPD upper bound. Users may choose to compute the equal-tailed credible intervals (CI) instead of the HPD intervals by setting 'HPD=FALSE'.

```
R> summary(fit_ma, HPD = TRUE, level = 0.95)
Call:
bmeta_analyze(formula = formula(f_1), data = cholesterol,
  prior = list(model = "NoRecovery"),
  mcmc = list(ndiscard = 1000, nkeep = 1000), control = list(scale_x = TRUE,
        verbose = TRUE))
Fixed-effects:
                       Post.Mean  Std.Dev  HPD(Lower)  HPD(Upper)
bldlc_1                   0.1394   0.0938     -0.0532      0.3051
bhdlc_1                  -0.5146   0.6329     -1.7641      0.7243
btg_1                     0.0018   0.0818     -0.1693      0.1607
age_1                     0.3785   0.4389     -0.5308      1.1648
durat_1                   0.9699   0.5234     -0.1658      1.9979
white_1                  -5.2449   7.4048    -19.7345      8.9764
male_1                   -1.2652  12.3948    -25.6807     23.0538
dm_1                      0.3987   5.3392    -10.4995      9.4673
bldlc_2                  -0.0128   0.0137     -0.0393      0.0134
bhdlc_2                  -0.0294   0.1280     -0.2887      0.2149
btg_2                     0.0228   0.0212     -0.0203      0.0619
age_2                    -0.0963   0.0735     -0.2422      0.0460
durat_2                  -0.0007   0.0712     -0.1320      0.1382
white_2                   5.5822   1.3044      3.0802      8.1901
male_2                   -2.1449   2.6956     -7.5975      2.6164
dm_2                     -1.0457   1.0923     -3.1816      1.0341
bldlc_3                   0.0141   0.0406     -0.0719      0.0853
bhdlc_3                   0.0082   0.2637     -0.5278      0.4872
btg_3                    -0.0734   0.0467     -0.1598      0.0154
age_3                    -0.1385   0.2043     -0.5077      0.2591
durat_3                   0.0996   0.2546     -0.3718      0.6080
white_3                  -0.7053   5.2345    -11.0570      8.9591
male_3                   14.5482   7.1735      0.0642     28.2959
dm_3                      5.0535   2.9542     -1.4157     10.5545
(Intercept)*(1-2nd)_1   -42.8675   3.2934    -48.8356    -35.8794
treat*(1-2nd)_1         -12.1435   1.1211    -14.4040    -10.0135
(Intercept)*2nd_1        -3.2219   3.0426     -9.3109      2.7419
treat*2nd_1             -20.0843   1.5235    -23.3169    -17.2637
(Intercept)*(1-2nd)_2     5.1425   0.5840      3.8240      6.1036
treat*(1-2nd)_2           2.0787   0.4718      1.0663      3.0222
(Intercept)*2nd_2         0.7346   0.5814     -0.3377      1.8738
treat*2nd_2               1.3482   0.3116      0.7579      1.9880
(Intercept)*(1-2nd)_3   -18.5035   2.1543    -22.3232    -13.7985
treat*(1-2nd)_3          -4.7726   0.9952     -6.7784     -2.9511
(Intercept)*2nd_3        -4.1805   1.2910     -6.7672     -1.7005
treat*2nd_3              -8.8579   0.9443    -10.6208     -7.0530
-----------------------------------------------------
*HPD level:  0.95
```

The suffixed '_j' where j can be 1, 2, or 3 corresponds to the response endpoint. Since scale_x = TRUE is equivalent to scale(<var>,center=TRUE,scale=TRUE), the covariates have been centered, which affects the interpretation of the intercepts. This allows us to interpret (Intercept*(1-2nd)_1)=-42.8675 as the statin effect in the first-line studies, where 2nd represents the indicator variable for second-line studies evaluating to one if second-line and zero otherwise. On the other hand, the coefficient estimate -12.1435 for treat*(1-2nd)_1 is the Statin+Ezetimibe effect, compared to administering statin alone. For the second-line studies where patients had already been on statin, (Intercept)*2nd_1=-3.2219 came out insignificant, according to the 95% HPD interval, as anticipated because the treatment for this group was merely a continuation of taking statin. The coefficient estimate -20.0843 for treat*2nd_1 shows that ezetimibe on top of statin has a greater cholesterol-lowering effect than statin alone.

print is similar to summary but additionally prints the model specification. The output from

'print(fit_ma)' is given as follows.

```
R> print(fit_ma, HPD = TRUE, level = 0.95)
Call:
bmeta_analyze(formula = formula(f_1),
    data = cholesterol, prior = list(model = "NoRecovery"),
    mcmc = list(ndiscard = 1000, nkeep = 1000),
    control = list(scale_x = TRUE, verbose = TRUE))
Model:
  (Aggregate mean)
    y_kt = X_kt * theta + W_kt * gamma_k + N(0, Sigma_kt / n_kt)
  (Sample Variance)
    (n_kt - 1) S_kt ~ Wishart(n_kt - 1, Sigma_kt)
  (Random effects)
    [gamma_k | Omega] ~ N(0, Omega)
Priors:
   theta ~ MVN(0,  1e+05  * I_p)
   Omega_j^{-1} ~ Wishart( 2.1 , Omega0)
   Sigma_kt = diag(sig_{tk,11}^2, ..., sig_{tk,JJ}^2)
   where sig_{tk,jj}^2 ~ IG( 0.1 ,  3.1 )
--------------------------------------------------
Number of trials:      26
Number of arms:        52
Number of treatments:  2
                      Post.Mean  Std.Dev  HPD(Lower)  HPD(Upper)
bldlc_1                  0.1394   0.0938     -0.0532      0.3051
bhdlc_1                 -0.5146   0.6329     -1.7641      0.7243
btg_1                    0.0018   0.0818     -0.1693      0.1607
age_1                    0.3785   0.4389     -0.5308      1.1648
durat_1                  0.9699   0.5234     -0.1658      1.9979
white_1                 -5.2449   7.4048    -19.7345      8.9764
male_1                  -1.2652  12.3948    -25.6807     23.0538
dm_1                     0.3987   5.3392    -10.4995      9.4673
bldlc_2                 -0.0128   0.0137     -0.0393      0.0134
bhdlc_2                 -0.0294   0.1280     -0.2887      0.2149
btg_2                    0.0228   0.0212     -0.0203      0.0619
age_2                   -0.0963   0.0735     -0.2422      0.0460
durat_2                 -0.0007   0.0712     -0.1320      0.1382
white_2                  5.5822   1.3044      3.0802      8.1901
male_2                  -2.1449   2.6956     -7.5975      2.6164
dm_2                    -1.0457   1.0923     -3.1816      1.0341
bldlc_3                  0.0141   0.0406     -0.0719      0.0853
bhdlc_3                  0.0082   0.2637     -0.5278      0.4872
btg_3                   -0.0734   0.0467     -0.1598      0.0154
age_3                   -0.1385   0.2043     -0.5077      0.2591
durat_3                  0.0996   0.2546     -0.3718      0.6080
white_3                 -0.7053   5.2345    -11.0570      8.9591
male_3                  14.5482   7.1735      0.0642     28.2959
dm_3                     5.0535   2.9542     -1.4157     10.5545
(Intercept)*(1-2nd)_1  -42.8675   3.2934    -48.8356    -35.8794
treat*(1-2nd)_1        -12.1435   1.1211    -14.4040    -10.0135
(Intercept)*2nd_1       -3.2219   3.0426     -9.3109      2.7419
treat*2nd_1            -20.0843   1.5235    -23.3169    -17.2637
(Intercept)*(1-2nd)_2    5.1425   0.5840      3.8240      6.1036
treat*(1-2nd)_2          2.0787   0.4718      1.0663      3.0222
(Intercept)*2nd_2        0.7346   0.5814     -0.3377      1.8738
treat*2nd_2              1.3482   0.3116      0.7579      1.9880
(Intercept)*(1-2nd)_3  -18.5035   2.1543    -22.3232    -13.7985
treat*(1-2nd)_3         -4.7726   0.9952     -6.7784     -2.9511
(Intercept)*2nd_3       -4.1805   1.2910     -6.7672     -1.7005
treat*2nd_3             -8.8579   0.9443    -10.6208     -7.0530
--------------------------------------------------
*HPD level:  0.95
```

For model comparison, the deviance information criterion (DIC) and the logarithm of the pseudo

marginal likelihood (LPML) can be computed using the `model_comp` method. The DIC will also contain $\text{Dev}(\bar{\boldsymbol{\eta}})$ and p_D. Similarly, the LPML will contain the logarithm of the CPOs, which is omitted.

```
R> dic <- model_comp(fit_ma, "dic")
R> lpml <- model_comp(fit_ma, "lpml")
R> c(dic$dic, dic$Dev, dic$pD)
[1] 827.80726 734.50691  46.65018

R> lpml$lpml
[1] -428.1813
```

Network meta-analysis

metapack includes another data set, 'TNM', which consists of 29 studies, dubbed the *Triglycerides Network Meta (TNM)* data (Li et al., 2019). The data set has 73 observations and 15 variables, which can be loaded via '`data("TNM")`'. The aggregate response variable is the mean percentage difference in triglycerides ('ptg'), paired with its corresponding standard deviation ('sdtg'). Similarly to

Variable	Description
`trial`	trial identifier
`treat`	treatment indicator for placebo (PBO), simvastatin (S), atorvastatin (A), lovastatin (L), rosuvastatin (R), pravastatin (P), ezetimibe (E), simvastatin+ezetimibe (SE), atorvastatin+ezetimibe (AE), lovastatin+ezetimibe (LE), or pravastatin+ezetimibe (PE)
`n`	the number of participants in the study arms
`ptg`	percentage change from baseline in triglycerides (TG)
`sdtg`	sample standard deviation of percentage change in triglycerides (TG)
`bldlc`	baseline LDL-C
`bhdlc`	baseline HDL-C
`btg`	baseline triglycerides (TG)
`age`	age in years
`white`	the proportion of white participants
`male`	the proportion of male participants
`bmi`	body fat index
`potencymed`	the proportion of medium statin potency
`potencyhigh`	the proportion of high statin potency
`durat`	duration in weeks

Table 4: Variables included in the `TNM` data set.

'`cholesterol`', variable descriptions are also available through `help("TNM")`.

LPML in Section Model comparison is not the only quantity affected by the treatments only included in a single trial. The variances of the corresponding treatment effects are nonestimable. Li et al. (2019) proposes to group those treatments and allow the treatments in a group to share the same variance. This grouping scheme can be easily achieved using `match` in R:

```
R> TNM$group <- factor(match(TNM$treat, c("PBO", "R"), nomatch = 0))
```

In the following demonstration, we consider the following model:

```
R> f_2 <- 'ptg | sdtg ~
+     0 + bldlc + bhdlc + btg + age + white + male + bmi +
+     potencymed + potencyhigh + durat + ns(n) |
+     scale(bldlc) + scale(btg) + group | treat  + trial'
```

The model can be fit by running

```
R> set.seed(2797542)
R> fit_nma <- bmeta_analyze(formula(f_2), data = TNM,
+  mcmc = list(ndiscard = 1000, nskip = 1, nkeep = 1000),
+  control=list(scale_x = TRUE, verbose=TRUE))
```

Again, the model summary can be obtained using either `summary` or `print` with minor differences.

```
R> summary(fit_nma)
Call:
bmeta_analyze(formula = formula(f_2), data = TNM, mcmc = list(ndiscard = 1000,
    nskip = 1, nkeep = 1000), control = list(scale_x = TRUE,
    verbose = TRUE))

Posterior inference in network meta-regression models
Fixed-effects:
             Post.Mean  Std.Dev  HPD(Lower)  HPD(Upper)
bldlc          -0.0071   0.0171     -0.0409      0.0230
bhdlc           0.2332   0.2490     -0.2802      0.7027
btg             0.0846   0.0366      0.0006      0.1460
age            -0.0068   0.1028     -0.2003      0.2020
white          -7.2148   1.9867    -10.7112     -3.0757
male            1.2323   7.3305    -12.5660     15.6644
bmi            -0.4505   0.3534     -1.1544      0.2184
potencymed      6.8278   7.9203     -8.5359     22.4551
potencyhigh    -0.6474   7.9330    -16.9216     14.4191
durat           0.1880   0.1879     -0.1638      0.5652
A             -24.3175   1.7820    -27.9255    -20.9635
AE            -30.3604   2.9591    -35.7474    -24.2297
E              -5.2737   6.1207    -17.4431      6.4066
L             -11.6521   4.3747    -20.1106     -2.7431
LE            -26.9507   3.4730    -33.3236    -19.9223
P              -8.3357   4.5065    -16.9311      0.7870
PBO             1.6170   6.0629     -9.9366     13.6298
PE            -22.7722   3.6333    -29.6129    -15.7540
R             -17.7669   2.0010    -21.3989    -13.3839
S             -19.1616   1.2175    -21.7647    -16.9279
SE            -21.8271   2.0509    -25.8031    -17.6347
--------------------------------------------------
*HPD level:  0.95
```

We observe that with covariate adjustment, all active treatments (A, AE, E, L, LE, P, PE, R, S, SE) reduce triglyceride (TG) more effectively than the placebo (PBO), although E and P have 95% HPD intervals including zero.

The output from '`print(fit_nma)`' further includes the model specification, and summary statistics for $\boldsymbol{\phi}$.

```
R> print(fit_nma)
Call:
bmeta_analyze(formula = formula(f_2), data = TNM, mcmc = list(ndiscard = 1000,
    nskip = 1, nkeep = 1000), control = list(scale_x = TRUE,
    verbose = TRUE))
Model:
  (Aggregate mean)
    y_kt = x_kt'theta + tau_kt * gamma_kt + N(0, sigma_kt^2 / n_kt)
  (Sample Variance)
    (n_kt - 1) S^2 / sigma_kt^2 ~ chi^2(n_kt - 1)
  (Random effects)
    [gam | Rho,nu] ~ MVT(0, E_k' Rho E_k, nu)
Priors:
  theta      ~ MVN(0, c01 * I_p), c01= 1e+05
  phi        ~ MVN(0, c02 * I_q), c02= 4
  p(sigma^2) ~ 1/sigma^2 * I(sigma^2 > 0)
  p(Rho)     ~ 1
--------------------------------------------------
Number of studies:     29
Number of arms:        73
Number of treatments:  11
             Post.Mean  Std.Dev  HPD(Lower)  HPD(Upper)
bldlc          -0.0071   0.0171     -0.0409      0.0230
bhdlc           0.2332   0.2490     -0.2802      0.7027
```

```
btg              0.0846   0.0366      0.0006      0.1460
age             -0.0068   0.1028     -0.2003      0.2020
white           -7.2148   1.9867    -10.7112     -3.0757
male             1.2323   7.3305    -12.5660     15.6644
bmi             -0.4505   0.3534     -1.1544      0.2184
potencymed       6.8278   7.9203     -8.5359     22.4551
potencyhigh     -0.6474   7.9330    -16.9216     14.4191
durat            0.1880   0.1879     -0.1638      0.5652
A              -24.3175   1.7820    -27.9255    -20.9635
AE             -30.3604   2.9591    -35.7474    -24.2297
E               -5.2737   6.1207    -17.4431      6.4066
L              -11.6521   4.3747    -20.1106     -2.7431
LE             -26.9507   3.4730    -33.3236    -19.9223
P               -8.3357   4.5065    -16.9311      0.7870
PBO              1.6170   6.0629     -9.9366     13.6298
PE             -22.7722   3.6333    -29.6129    -15.7540
R              -17.7669   2.0010    -21.3989    -13.3839
S              -19.1616   1.2175    -21.7647    -16.9279
SE             -21.8271   2.0509    -25.8031    -17.6347
phi1             0.4088   0.2716     -0.1610      0.8882
phi2            -0.3248   0.3869     -1.1721      0.2902
phi3             0.2692   0.2239     -0.1859      0.6835
phi4            -1.0862   1.2024     -3.2686      0.9419
phi5             0.5973   0.3407     -0.0341      1.2829
-----------------------------------------------------
*HPD level:  0.95
```

The model comparison measures are computed using `model_comp`. For example,

```
R> dic <- model_comp(fit_nma, "dic")
R> c(dic$dic, dic$Dev, dic$pD)
[1] 386.41450 334.72118  25.84666

R> lpml <- model_comp(fit_nma, "lpml")
R> lpml$lpml
[1] -161.518
```

The `plot` method will generate trace plots and density plots of the fixed-effect coefficients. Figure 2 shows the trace plots and density plots of treatments R, S, and SE from the network meta-analysis model, generated by '`plot(fit_nma)`'.

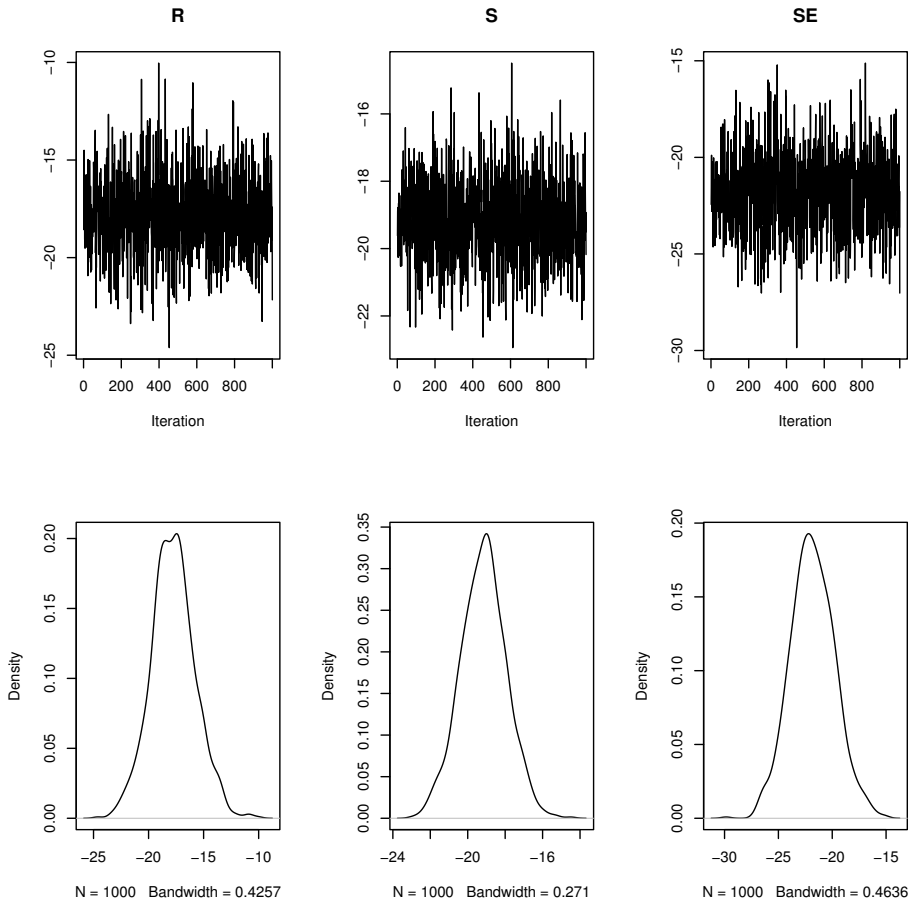


Figure 2: The trace plots and density plots of treatments R, S, and SE from the network meta-analysis model generate from '`plot(fit_nma)`'. The trace plots show good mixing and convergence of MCMC chains and the density plots indicate that the marginal posterior distribution for each treatment effect is roughly symmetric. The treatment labels come from the '`group`'. If '`group`' is a factor, its levels will be used. Otherwise, treatment labels will be numbered.

In addition to the MCMC diagnostics, network meta models can be visualized using SUCRA plots, i.e. '`plot(sucra(fit_nma))`'. Figure 3 shows the SUCRA plot from the '`fit_nma`' object. The `plot` function for SUCRA uses **ggplot2** (Wickham, 2016) and **gridExtra** (Auguie, 2017) to generate and combine plots.

7 Discussion

This paper introduces **metapack** for (network) meta-analysis, and illustrates the usage of the main function, `bmeta_analyze`. We further demonstrate how to analyze data using the `cholesterol` and `TNM` data sets included in **metapack**. The package relies on **Rcpp**, **RcppArmadillo**, and **OpenMP** to boost computation speed. Furthermore, we propose a unified formula structure to represent meta-analytic data using **Formula**, which we hope to see gain currency in the community.

There is a cautionary remark worth mentioning about the ratio between the correlation information in the data and the number of correlation-related parameters. The number of endpoints and the number of arms in the multivariate meta-analysis models are critical in determining whether the missing correlations ($R_{kt}, \boldsymbol{\rho}$) are identifiable. If there are too many endpoints, there must be enough data points. Otherwise, the prior distribution for $\boldsymbol{\rho}$ cannot be noninformative. From our experience, using only one of the two therapies in `cholesterol` results in nonidentifiable correlations since each off-diagonal entry of $\boldsymbol{\rho}$ will have four observations on average. This can render the MCMC algorithm unstable, covering the whole $(-1, 1)$. This could potentially break the MCMC chain if any of the elements gets too close to either 1 or -1, violating positive definiteness.

The efficient estimation of the correlation matrix is an important future research direction for which the package will serve as a valuable repository of resources. The first few future implementations will

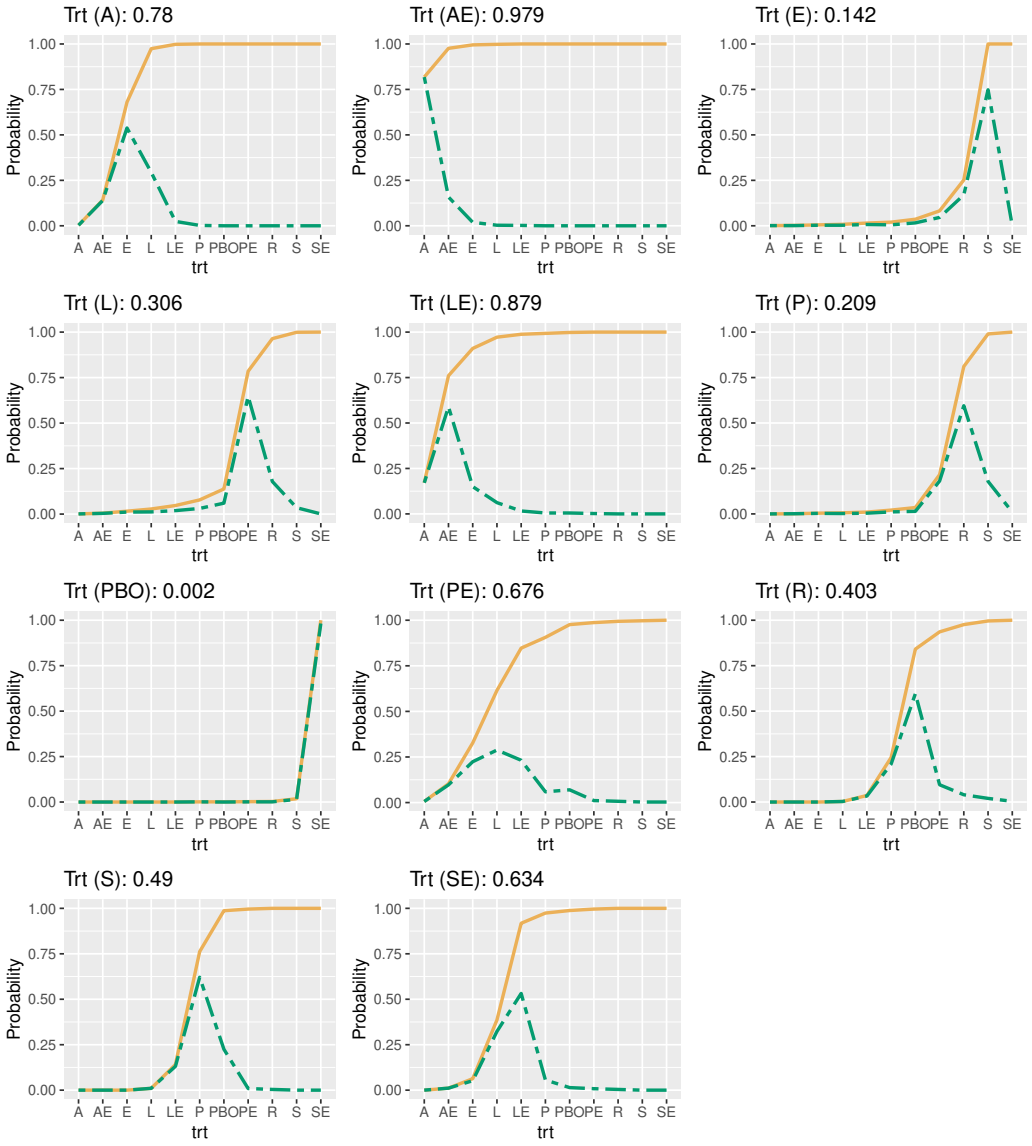


Figure 3: The SUCRA plot for all treatment arms generated by `plot(sucra(fit_nma))`. The green dashed line represents the discrete probability mass and the orange solid line represents the cumulative probability. The SUCRA values are displayed on top of each subplot. For optimal visualization, we recommend the labels be three characters or fewer. AE is ranked highest according to SUCRA, followed by LE.

focus on the regression modeling of the correlations to address the cases where either the data are too small to estimate the correlations or the number of treatments is too large. Models accommodating various circumstances regarding the sample variances are under active development. For example, researchers might want to suppress the sampling of the variance-covariance matrix as a whole for various reasons. Researchers might also have partially observed *sample variances*, not covariances, depending on the study included in the systematic review. **metapack** in the coming releases will provide these options. Therefore, **metapack** has great potential for further development.

Acknowledgments

We would like to thank the Editor, the Associate Editor, and the two reviewers for their helpful comments and suggestions, which led to a much improved version of the paper. Dr. Chen and Dr. Ibrahim's research was partially supported by NIH grants #GM70335 and #P01CA142538, and Merck & Co., Inc., Rahway, NJ, USA. Dr. Kim's research was supported by the Intramural Research Program of National Institutes of Health, National Cancer Institute.

Bibliography

B. Auguie. **gridExtra**: *Miscellaneous Functions for "Grid" Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p157]

D. H. Bailey, K. Jeyabalan, and X. S. Li. A comparison of three high-precision quadrature schemes. *Experimental Mathematics*, 14(3):317–329, 2005. doi: em/1128371757. [p149]

S. Balduzzi, G. Rücker, and G. Schwarzer. How to perform a meta-analysis with R: a practical tutorial. *Evidence-based mental health*, 22(4):153–160, 2019. [p142]

C. S. Berkey, D. C. Hoaglin, F. Mosteller, and G. A. Colditz. A random-effects regression model for meta-analysis. *Statistics in medicine*, 14(4):395–411, 1995. [p142]

J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West. Non-centered parameterisations for hierarchical models and data augmentation. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, volume 307. Oxford University Press, USA, 2003. [p146]

M. Borenstein, L. V. Hedges, J. P. Higgins, and H. R. Rothstein. *Introduction to meta-analysis*. John Wiley & Sons, 2011. [p142]

I. Chalmers, L. V. Hedges, and H. Cooper. A brief history of research synthesis. *Evaluation & the health professions*, 25(1):12–37, 2002. [p142]

R. DerSimonian and N. Laird. Meta-analysis in clinical trials. *Controlled clinical trials*, 7(3):177–188, 1986. [p142]

T. Ding and G. Baio. **bmeta**: *Bayesian Meta-Analysis and Meta-Regression*, 2016. URL https://CRAN.R-project.org/package=bmeta. R package version 0.1.2. [p142]

D. Eddelbuettel and J. J. Balamuta. Extending R with C++: A Brief Introduction to **Rcpp**. *PeerJ Preprints*, 5:e3188v1, aug 2017. ISSN 2167-9843. URL https://doi.org/10.7287/peerj.preprints.3188v1. [p145]

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://dx.doi.org/10.1016/j.csda.2013.02.005. [p145]

D. Eddelbuettel, J. W. Emerson, and M. J. Kane. **BH**: *Boost C++ Header Files*, 2019. URL https://CRAN.R-project.org/package=BH. R package version 1.69.0-1. [p149]

A. Gasparrini, B. Armstrong, and M. G. Kenward. Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*, 31(29):3821–3839, 2012. [p142]

B. K. Guenhan. *MetaStan: Bayesian Meta-Analysis via 'Stan'*, 2020. URL https://CRAN.R-project.org/package=MetaStan. R package version 0.2.0. [p142]

B. K. Guenhan, T. Friede, and L. Held. A design-by-treatment interaction model for network meta-analysis and meta-regression with integrated nested Laplace approximations. *Research Synthesis Methods*, pages 179–194, 2018. URL https://doi.org/10.1002/jrsm.1285. [p142]

A. Guolo. Higher-order likelihood inference in meta-analysis and meta-regression. *Statistics in Medicine*, (31):313–327, 2012. [p142]

J. Hartung, G. Knapp, and B. K. Sinha. *Statistical meta-analysis with applications*, volume 738. John Wiley & Sons, 2011. [p142]

H. M. Huizenga, I. Visser, and C. V. Dolan. Hypothesis testing in random effects meta-regression. *British journal of mathematical and statistical psychology*, 64:1–19, 2011. [p142]

H. Li, M.-H. Chen, J. G. Ibrahim, S. Kim, A. K. Shah, J. Lin, and A. M. Tershakovec. Bayesian inference for network meta-regression using multivariate random effects with applications to cholesterol lowering drugs. *Biostatistics*, 20(3):499–516, 2019. [p154]

H. Li, D. Lim, M.-H. Chen, J. G. Ibrahim, S. Kim, A. K. Shah, and J. Lin. Bayesian network meta-regression hierarchical models using heavy-tailed multivariate random effects with covariate-dependent variances. *Statistics in Medicine*, 2021. [p143]

T. Lumley. **rmeta***: Meta-Analysis*, 2018. URL https://CRAN.R-project.org/package=rmeta. R package version 3.0. [p142]

OpenMP Architecture Review Board. OpenMP application programming interface version 5.0, 11 2018. URL https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf. [p149]

M. Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, 2003. [p142]

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://journal.r-project.org/archive/. [p149]

C. Röver. Bayesian random-effects meta-analysis using the bayesmeta R package. *Journal of Statistical Software*, 4 2020. URL https://doi.org/10.18637/jss.v093.i06. [p142]

G. Schwarzer. **meta**: An R package for meta-analysis. *R News*, 7(3):40–45, 2007. [p142]

I. M. Skovgaard et al. An explicit large-deviation approximation to one-parameter tests. *Bernoulli*, 2(2): 145–165, 1996. [p142]

B. J. Smith. boa: An R package for MCMC output convergence assessment and posterior inference. *Journal of Statistical Software*, 21(11):1–37, 2007. [p149]

D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the royal statistical society: Series b (statistical methodology)*, 64(4):583–639, 2002. [p148]

Stan Development Team. RStan: the R interface to Stan, 2020. URL http://mc-stan.org/. R package version 2.21.1. [p142]

S. Sturtz, U. Ligges, and A. Gelman. R2winbugs: A package for running WinBUGS from R. *Journal of Statistical Software*, 12(3):1–16, 2005. URL http://www.jstatsoft.org. [p142]

H. Takahasi and M. Mori. Double exponential formulas for numerical integration. *Publications of the Research Institute for Mathematical Sciences*, 9(3):721–741, 1974. [p149]

U.S. Food and Drug Administration, Center for Drug Evaluation, and Research and Center for Biologics Evaluation and Research. *Meta-Analyses of Randomized Controlled Clinical Trials to Evaluate the Safety of Human Drugs or Biological Products*. U.S. Food and Drug Administration, 2018. [p142]

W. Viechtbauer. Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, 36(3):1–48, 2010. URL http://www.jstatsoft.org/v36/i03/. [p142]

H. Wickham. **ggplot2***: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p157]

H. Yao, S. Kim, M.-H. Chen, J. G. Ibrahim, A. K. Shah, and J. Lin. Bayesian inference for multivariate meta-regression with a partially observed within-study sample covariance matrix. *Journal of the American Statistical Association*, 110(510):528–544, 2015. [p143, 150]

A. Zeileis and Y. Croissant. Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software*, 34(1):1–13, 2010. URL https://doi.org/10.18637/jss.v034.i01. [p145]

Daeyoung Lim
University of Connecticut
215 Glenbrook Rd. U-4120 Storrs, CT 06269-4120
United States
daeyoung.lim@uconn.edu

Ming-Hui Chen
University of Connecticut
215 Glenbrook Rd. U-4120 Storrs, CT 06269-4120
United States
ming-hui.chen@uconn.edu

Joseph G. Ibrahim
University of North Carolina
3109 McGavran-Greenberg Hall CB #7420 Chapel Hill, NC 27599
United States
ibrahim@bios.unc.edu

Sungduk Kim
Biostatistics Branch
Division of Cancer Epidemiology & Genetics
National Cancer Institute
National Institutes of Health
9609 Medical Center Drive Rockville, MD 20852
United States
kims2@mail.nih.gov

Arvind K. Shah
Merck & Co., Inc.
126 East Lincoln Avenue, P.O. Box 2000, Rahway, NJ 07065
United States
arvind_shah@merck.com

Jianxin Lin
Merck & Co., Inc.
126 East Lincoln Avenue, P.O. Box 2000, Rahway, NJ 07065
United States
jianxin_lin@merck.com

did2s: Two-Stage Difference-in-Differences

by Kyle Butts and John Gardner

Abstract Recent work has highlighted the difficulties of estimating difference-in-differences models when the treatment is adopted at different times for different units. This article introduces the R package **did2s** which implements the estimator introduced in Gardner (2022). The article provides an approachable review of the underlying econometric theory and introduces the syntax for the function `did2s`. Further, the package introduces functions, event_study and plot_event_study, which uses a common syntax to implement all of the modern event-study estimators.

Introduction

A rapidly growing econometric literature has identified difficulties in traditional difference-in-differences estimation when treatment turns on at different times for different groups and when the effects of treatment vary across groups and over time (Callaway and Sant'Anna 2020; Sun and Abraham 2020; Goodman-Bacon 2018; Borusyak, Jaravel, and Spiess 2021; Chaisemartin and D'Haultfoeuille 2019). Gardner (2022) proposes an estimator of the two-way fixed-effects model that is quick and intuitive. The estimator relies on the standard two-way fixed-effect model (see the following section) and forms an intuitive estimate: the average difference in outcomes between treated and untreated units after removing fixed unit- and time-invariant shocks.

This article first discusses the modern difference-in-differences theory in an approachable way and second discusses the software package, **did2s**, which implements the two-stage estimation approach proposed by Gardner (2022) to estimate robustly the two-way fixed-effects (TWFE) model. There are two notable technical features of this package. First, `did2s` utilizes the incredibly fast package, **fixest** (Bergé 2018), which can estimate regressions with a high number of fixed-effects very quickly. Second, since there are a few alternative TWFE event-study estimators implemented in `R`, each with their own syntax and data formatting requirements, the package also has a set of functions that allow quick estimation and plotting of every alternative event study estimator using a standardized syntax. This allows for easy comparison between the results of different methods.

Difference-in-differences theory

Researchers commonly use the difference-in-differences (DiD) methodology to estimate the effects of treatment in the case where treatment is non-randomly assigned. Instead of random assignment giving rise to identification, the DiD method relies on the so-called "parallel trends" assumption, which asserts that outcomes would evolve in parallel between the treated and untreated groups *in a world where the treated were untreated*. This is formalized with the *two-way fixed-effects* (TWFE) model. In a static setting where treatment effects are *constant* across treatment groups and over time, researchers estimate the **static TWFE model**:

$$y_{igt} = \mu_g + \eta_t + \tau D_{gt} + \varepsilon_{igt}, \tag{1}$$

where y_{igt} is the outcome variable of interest, i denotes the individual, t denotes time, and g denotes group membership where a "group" is defined as all units that start treatment at time g.[1] μ_g is a vector of time-invariant group fixed-effects, η_t is a vector of shocks in a given time period that is experienced by all individuals equally, and D_{gt} is an indicator for whether initial-treatment group g is receiving treatment in period t, i.e. $D_{gt} \equiv \mathbb{1}(g \leq t)$. The coefficient of interest is τ, which is the ***(constant)* average effect of the treatment on the treated** (ATT). If it is indeed true that the treatment effect is constant across groups and over time, then the estimate formed by estimating the static TWFE model will be consistent for τ under a parallel trends assumption on the error term.

However, treatment effects are not constant in most settings. The magnitude of a unit's treatment effect can differ based on group status g (e.g. if groups that benefit more from a policy implement it earlier) and treatment duration (e.g. if treatment effects grow as the policy has been in place for longer periods). Therefore to enrich our model, we allow heterogeneity in treatment effects across g and t by introducing the **group-time average treatment effect**, τ_{gt}. Correspondingly, we modify the TWFE model as follows:

$$y_{igt} = \mu_g + \eta_t + \tau_{gt} D_{gt} + \varepsilon_{igt}.$$

[1] In the literature, never treated units often are given a value of $g = \infty$.

The key difference is that treatment effects are allowed to differ based on group status g and time period t. Estimating any individual τ_{gt} may not be desirable since there would be too few observations. Instead, researchers aggregate group-time average treatment effects into the **overall average treatment effect**, τ, which averages across τ_{gt}:

$$\tau \equiv \sum_{g,t} \frac{N_{gt}}{N_{post}} \tau_{gt},$$

where N_{gt} denotes the number of observations in (g,t) and N_{post} is the number of post-treatment observations ($t \geq g$). The natural question is, "does the static TWFE model, (1), produce a consistent estimate for the overall average treatment effect?" Except for a few specific scenarios, the answer is no (Sun and Abraham 2020; Goodman-Bacon 2018; Borusyak, Jaravel, and Spiess 2021; Chaisemartin and D'Haultfoeuille 2019).

One way of thinking about this disappointing result is through the Frisch–Waugh–Lovell (FWL) theorem (Frisch and Waugh 1933). This theorem says that estimating the Static TWFE model is equivalent to estimating

$$y_{igt} - \hat{\mu}_g - \hat{\eta}_t = \tau \tilde{D}_{gt} + \tilde{\varepsilon}_{gt},$$

where $\tilde{D}_{gt}$ denotes the residuals from regressing D_{gt} on μ_g and η_t; $\hat{\mu}_g$ and $\hat{\eta}_t$ are estimates for the group and time fixed-effects, respectively. The left-hand side of this equation, under a parallel trends restriction on the error term ε_{it}, is our estimate for τ_{gt}. Therefore, the FWL theorem tells us estimating the static TWFE model is equivalent to estimating[2]

$$\hat{\tau}_{gt} = \tau \tilde{D}_{gt} + \tilde{\varepsilon}_{gt}$$

The resulting estimate for τ can be written as:

$$\hat{\tau} \equiv \sum_{g,t} w_{gt} \hat{\tau}_{gt},$$

where w_{gt} is the weight put on the corresponding $\hat{\tau}_{gt}$. Results of Gardner (2022), Borusyak, Jaravel, and Spiess (2021), and Chaisemartin and D'Haultfoeuille (2019) all characterize the weights w_{gt} from this regression. There are only two cases where the $\hat{\tau}$ is a consistent estimate for the overall average treatment effect. First, when treatment occurs at the *same time* for all treated units, then w_{gt} is equal to N_{gt}/N_{post} for all $\{g,t\}$ and therefore $\hat{\tau}$ is a consistent estimate for the overall average treatment effect. The other scenario where $\hat{\tau}$ estimates the overall average treatment effect is when τ_{gt} is constant across group and time, i.e. $\tau_{gt} = \tau$. Since the weights, w_{gt}, always sum to one, we have that $\hat{\tau} = \sum w_{gt} \hat{\tau}_{gt} \rightarrow \sum w_{gt} \tau = \tau$.

The above cases are not the norm in research. If there is heterogeneity in group-time treatment effects *and* units get treated at different times, then $\hat{\tau}$ is not a consistent estimate for the average treatment effect τ. Instead, $\hat{\tau}$ will be a weighted average of group-time treatment effects with some weights, w_{gt}, being potentially negative. This yields a treatment effect estimate that does not provide a good summary of the "average" treatment effect. It is even possible for the sign of $\hat{\tau}$ to differ from that of the overall average treatment effect. This would occur, for example, if negative weights are placed primarily on the largest (in magnitude) group-time treatment effects.

To summarize the modern literature, the fundamental problem faced in estimating the TWFE model is the potential negative weighting. The proposed methodology in Gardner (2022) is based on the fact that if $\hat{\tau}_{gt}$ is regressed on D_{gt}, instead of $\tilde{D}_{gt}$, the resulting weights would be exactly equal to N_{gt}/N_{post} and the coefficient of D_{gt} would estimate the overall average treatment effect.

Event-study estimates

Researchers have attempted to model treatment effect heterogeneity by allowing treatment effects to change over time. To do this, they introduce a (dynamic) event-study TWFE model:

$$y_{igt} = \mu_g + \eta_t + \sum_{k=-L}^{-2} \tau^k D_{gt}^k + \sum_{k=0}^{K} \tau^k D_{gt}^k + \varepsilon_{igt}, \tag{2}$$

where D_{gt}^k are lags/leads of treatment (k periods from initial treatment date). The coefficients of interests are the τ^k, which represent the average effect of being treated for k periods. For negative values of k, τ^k are known as "pre-trends," and represent the average deviation in outcomes for treated units k periods away from treatment, relative to their value in the reference period. These pre-trend

[2]This is a minor abuse of notation since $y_{igt} - \hat{\mu}_g - \hat{\eta}_t$ is an estimate for τ_{igt} which can be different from τ_{gt} if there is within group-time heterogeneity.

estimates are commonly used as a test of the parallel counterfactual trends assumption.

Our goal is to estimate the **average treatment effect of being exposed for k periods**, an average of τ_{gt} for only the set of $\{g,t\}$ where k periods have elapsed since g, i.e. $t-g=k$:

$$\tau^k = \sum_{g,t\,:\,t-g=k} \frac{N_{gt}^k}{N^k}\tau_{gt},$$

where the sum is over $\{g,t\}$ with $t-g=k$, N_{gt}^k is the number of observations in group g and N^k is the total number of observations with $t-g=k$. The results of Sun and Abraham (2020) show that even though we allow for our average treatment effects to vary over time τ^k, the negative weighting problems would arise if units are treated at different times *and* there is group-heterogeneity in treatment effects. Similar to the static TWFE model, the estimates of τ^k from running the event-study model form non-intuitively weighted averages of τ_{gt} with $w_{gt}^k \neq N_{gt}^k/N^k$. Even worse, the group-time treatment effects for $t-g\neq k$ will be included in the estimate of $\hat{\tau}^k$. Hence, the need for a robust difference-in-differences estimator remains even in the event-study model.

Two-stage difference-in-differences estimator

Gardner (2022) proposes an estimator to resolve the problem with the two-way fixed-effects approaches. Rather than attempting to estimate the group and time effects simultaneously with the ATT (causing D_{it} to be residualized), Gardner's approach proceeds from the observation that, under parallel trends, the group and time effects are identified from the subsample of untreated/not-yet-treated observations ($D_{gt}=0$). This suggests a simple two-stage difference-in-differences estimator:

1. Estimate the model
$$y_{igt} = \mu_g + \eta_t + \varepsilon_{igt},$$
using the subsample of untreated/not-yet-treated observations (i.e., all observations for which $D_{gt}=0$), retaining the estimated group and time effects to form the adjusted outcomes $\tilde{y}_{igt} \equiv y_{igt} - \hat{\mu}_g - \hat{\eta}_t$.
2. Regress adjusted outcomes $\tilde{y}_{igt}$ on treatment status D_{gt} or D_{gt}^k in the full sample to estimate treatment effects τ or τ^k.

To see why this procedure works, note that parallel trends implies that outcomes can be expressed as

$$\begin{aligned} y_{igt} &= \mu_g + \eta_t + \tau_{gt}D_{gt} + \varepsilon_{igt} \\ &= \mu_g + \eta_t + \bar{\tau}D_{gt} + (\tau_{gt}-\bar{\tau})D_{gt} + \varepsilon_{igt}, \end{aligned}$$

where $\tau_{gt} = E(Y_{igt}^1 - Y_{igt}^0 \mid g,t)$ is the average treatment effect for group g in period t[3] and $\bar{\tau} = E(\tau_{gt}|D_{gt}=1)$ is the overall average treatment effect[4]. Note from parallel trends, $E(\varepsilon_{igt}|D_{gt},g,t)=0$. Rearranging, this gives

$$y_{igt} - \mu_g - \eta_t = \bar{\tau}D_{gt} + (\tau_{gt}-\bar{\tau})D_{gt} + \varepsilon_{igt}.$$

Suppose you knew the time and group fixed-effects and were able to directly observe the left-hand side (later we will estimate the left-hand side). Regressing the adjusted y variable, on D_{gt} will produce a consistent estimator for $\bar{\tau}$. To see this, note that $E[(\tau_{gt}-\bar{\tau})D_{gt} \mid D_{gt}] = 0$. Hence, the treatment dummy is uncorrelated with the omitted variable and the average treatment effect is identified in the second-stage. Since we are not able to directly observe μ_g and η_t, we estimate them using the untreated/not-yet-treated observations in the first-stage. However, standard errors need adjustment to account for the added uncertainty from the first-stage estimation.

This approach can be extended to dynamic models by replacing the second stage of the procedure with a regression of residualized outcomes onto the leads and lags of treatment status, D_{gt}^k, $k \in \{-L,\ldots,K\}$. Under parallel trends, the second-stage coefficients on the lags identify the overall average effect of being treated for k periods (where the average is taken over all units treated for at least that many periods). The second-stage coefficients on the leads identify the average deviation from predicted counterfactual trends among units that are k periods away from treatment, which under parallel trends should be zero for any pre-treatment value of k. Hence, the coefficients on the leads represent a test of the validity of the parallel trends assumption.

[3] i.e., the average difference between treated and untreated potential outcomes y_{igt}^1 and y_{igt}^0, conditional on the observed treatment-adoption times.

[4] i.e., the population-weighted average of the group-time specific ATTs, τ_{gt}.

Inference

The standard variance-covariance matrix from the second-stage regression will be incorrect since it fails to account for the fact that the dependent variable is generated from the first-stage regression. However, this estimator takes the form of a joint generalized method of moments (GMM) estimator whose asymptotic variance is well understood (Newey and McFadden 1986).

Specifically, the estimator takes the form of a two-stage GMM estimator with the following two moment conditions:

$$m(\theta) = (Y - X_{10}'\gamma)X_{10}, \tag{3}$$

$$g(\gamma, \theta) = (Y - X_1'\gamma - X_2'\theta)X_2, \tag{4}$$

where X_1 is the matrix of group and time fixed-effects, X_{10} corresponds to the matrix X_1, but with rows corresponding to observations for which $D_{gt} = 1$ replaced with zeros (as only observations with $D_{gt} = 0$ are used in the first stage) and X_2 is the matrix of treatment variable(s). The first equation corresponds with the first stage and the second equation corresponds with the second stage. From Theorem 6.1 of Newey and McFadden (1986), the asymptotic variance of the two-stage estimator is

$$V = G_\theta^{-1} E\left[(g + G_\gamma \psi)(g + G_\gamma \psi)'\right] G_\theta^{-1'}, \tag{5}$$

where from our moment conditions, we have:

$$G_\theta = -E\left(X_2 X_2'\right),$$

$$G_\gamma = -E\left(X_2 X_1'\right),$$

$$\psi = E(X_{10} X_{10}')^{-1} \varepsilon_{10} X_{10}.$$

This can be estimated using

$$\left(X_2'X_2\right)^{-1} \left(\sum_{g=1}^{G} W_g' W_g\right) \left(X_2'X_2\right)^{-1}, \tag{6}$$

where

$$W_g = X_{2g}'\hat{\varepsilon}_{2g} - \hat{\varepsilon}_{10g}' X_{1g} \left(X_{1g}' X_{1g}\right)^{-1} \left(X_{1g}' X_{2g}\right),$$

and matrices indexed by g correspond to the gth cluster.

The did2s package

The did2s package introduces two sets of functions. The first is the `did2s` command which implements the two-stage difference-in-differences estimator as described above. The second is the `event_study` and `plot_event_study` commands that allow individuals to implement alternative 'robust' estimators using a singular common syntax.

The did2s command

The command `did2s` implements the two-stage difference-in-differences estimator following Gardner (2022). The general syntax is

```
did2s(data, yname, first_stage, second_stage,
      treatment, cluster_var, weights = NULL,
      bootstrap = FALSE, n_bootstraps = 250,
      verbose = TRUE)
```

and full details on the arguments is available in the help page, available by running `?did2s`. There are a few arguments that are worth discussing in more detail.

The `first_stage` and `second_stage` arguments require formula arguments. These formulas are passed to the `fixest::feols` function from fixest and can therefore utilize two non-standard formula options that are worth mentioning (Bergé 2018). First, fixed-effects can be inserted after the covariates, e.g. `~ x1 | fe_1 + fe_2`, which will make estimation much faster than using `factor(fe_1)`. Second, the function `fixest::i` can be used for treatment indicators instead of `factor`. The advantage of this is that you can easily specify the reference values, e.g. for event-study indicators where researchers typically want to drop time $t = -1$, `~ i(rel_year, ref = c(-1))` would be the correct second-stage

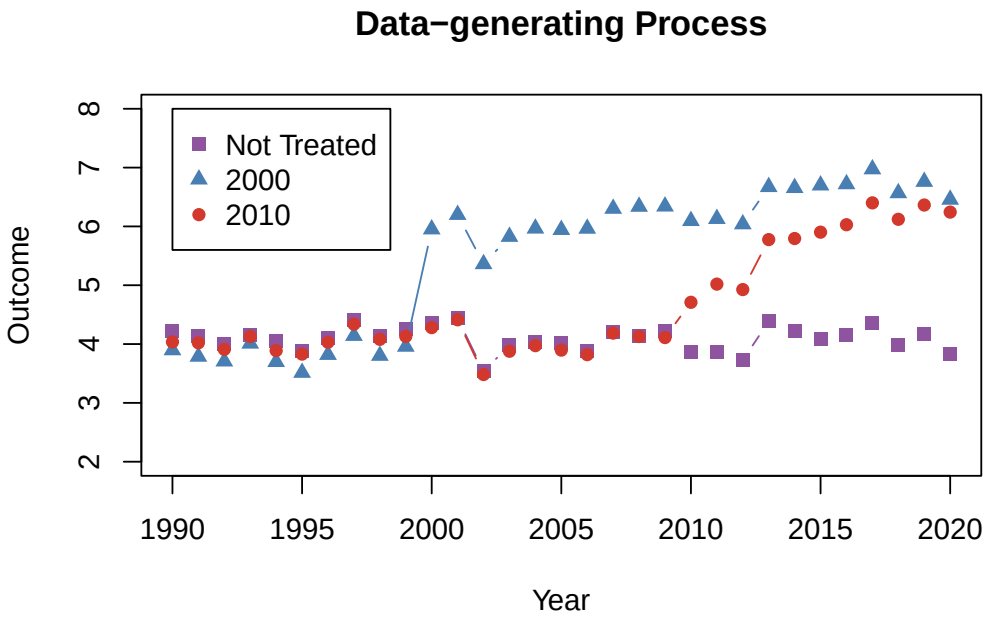


Figure 1: This figure plots simulated data with two treated groups and a never-treated group. Each line represents the average outcome (y-value) in a given year (x-value) for each of the three groups. In the absence of treatment, all three groups would exhibit parallel trends (staying around a value of 4 in each period). Each of the treated groups are experience different treatment effect magnitudes that grow over time. This treatment effect heterogeneity creates problems for the classical two-way fixed effect OLS estimator.

formula. Additionally, **fixest** has a number of post-estimation exporting commands to make tables with `fixest::etable` and event-study plots with `fixest::iplot/fixest::coefplot`. The `fixest::i` function is better integrated with these functions as we will see below.

The option `treatment` is the variable name of a $0/1$ variable that denotes when treatment is active for a given unit, D_{gt} in the above notation. Observations with $D_{gt} = 0$ will be used to estimate the first stage, which removes the problem of treatment effects contaminating estimation of the unit and time fixed-effects. However, as an important note, if you suspect anticipation effects before treatment begins, the `treatment` variable should be shifted forward by x periods for observations to prevent the aforementioned contamination. For example, if you suspect that units could experience treatment effects 1 period ahead of treatment (a so-called anticipatory effect), then the `treatment` should begin one period ahead. These anticipation effects can be estimated, after adjusting the `treatment` variable, by using a reference year of say, $t = -2$ and looking at the estimate for relative year -1.

Example usage of `did2s` For basic usage, I will use the simulated dataset, `df_het`, that comes with the **did2s** package with the command

```
data(df_het, package = "did2s")
```

The data-generating process is displayed in Figure 1. The lines represent the mean outcome for each treatment group and the never-treated group. In the absence of treatment, each group is simulated to be on parallel trends. There is heterogeneity in treatment effects both within a treatment group over time and across treatment groups.

First, we will calculate a static difference-in-differences estimate using the `did2s` function.

```
static = did2s(
    data = df_het,
    yname = "dep_var",
    treatment = "treat",
    first_stage = ~ 0 | unit + year,
    second_stage = ~ i(treat, ref = FALSE),
```

```
    cluster_var = "unit",
    verbose = FALSE
)

summary(static)

#> OLS estimation, Dep. Var.: dep_var
#> Observations: 46,500
#> Standard-errors: Custom
#>             Estimate Std. Error t value  Pr(>|t|)
#> treat::TRUE  2.23048   0.021408  104.19 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> RMSE: 1.0357   Adj. R2: 0.505683
```

Since the returning object is a `fixest` object, all the accompanying output commands from **fixest** are available to use. For example, we can create regression tables:

```
fixest::etable(static, fitstat = c("n"), tex = TRUE,
  title = "Estimate of Static TWFE Model",
 notes = "This table presents the estimated overall treatment effect. The effect is estimated using Two-Stage Diff
```

Table 1: Estimate of Static TWFE Model

Dependent Variable:	dep_var
Model:	(1)
Variables	
treat = TRUE	2.230***
	(0.0214)
Fit statistics	
Observations	46,500

Custom standard-errors in parentheses
*Signif. Codes: ***: 0.01, **: 0.05, *: 0.1*

This table presents the estimated overall treatment effect. The effect is estimated using Two-Stage Difference-in-Differences proposed by Gardner (2021). The estimated effect is close to the true value.

However, since there are dynamic treatment effects in this example, it is much better to estimate the dynamic effects themselves using an event-study specification. We will then plot the results using `fixest::iplot`, which plots coefficients corresponding to an `i()` variable. Note that `rel_year` is coded as `Inf` for never-treated units, so this has to be noted in the reference part of the formula.

```
es = did2s(
    data = df_het,
    yname = "dep_var",
    treatment = "treat",
    first_stage = ~ 0 | unit + year,
    second_stage = ~ i(rel_year, ref = c(-1, Inf)),
    cluster_var = "unit",
    verbose = FALSE
)

fixest::iplot(
    es,
    main = "Event study: Staggered treatment",
    xlab = "Relative time to treatment",
    col = "steelblue", ref.line = -0.5
)
```

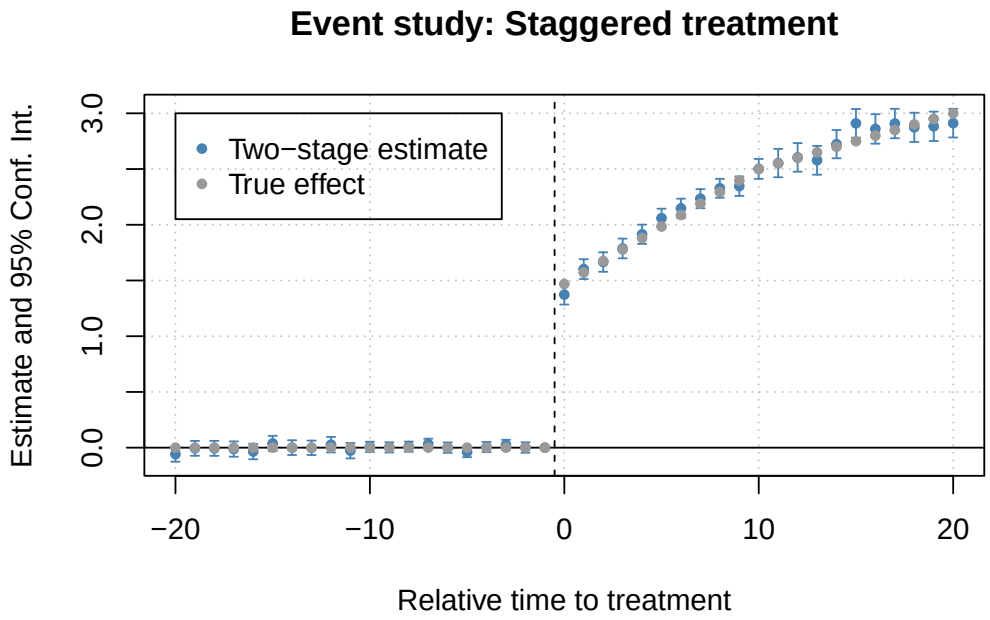


Figure 2: This figure plots the true treatment effect and estimates using the Two-Stage Difference-in-Differences proposed by Gardner (2021). The x-axis of this figure is the relative time to treatment, i.e. how many years pre-/post- treatment that period is. The y-axis is estimated treatment effects. There are two sets of points. The first is for the true effect which is equal to 0 in all pre-periods and in the post-period starts at 1.5 and linearly grows to 3 by post-period 20. The second set of points is the estimates from the two-stage difference-in-difference estimates which follows closely the true effects but with additional noise from estimation error.

```
# Add the (mean) true effects
true_effects = tapply((df_het$te + df_het$te_dynamic), df_het$rel_year, mean)
true_effects = head(true_effects, -1)
points(-20:20, true_effects, pch = 20, col = "grey60")

# Legend
legend(x=-20, y=3, col = c("steelblue", "grey60"),
       pch = c(20, 20),
       legend = c("Two-stage estimate", "True effect"))
```

The event study estimates are found in Figure 2 and match closely to the true average treatment effects. For comparison to traditional OLS estimation of the event-study specification, Figure 3 plots point estimates from both methods. As pointed out by Sun and Abraham (2020), treatment effect heterogeneity between groups biases the estimated pre-trends. In the figure below, the OLS estimates appear to show violations of pre-trends even though the data was simulated under parallel pre-trends.

```
twfe = feols(dep_var ~ i(rel_year, ref=c(-1, Inf)) | unit + year, data = df_het)

fixest::iplot(list(es, twfe), sep = 0.2, ref.line = -0.5,
      col = c("steelblue", "#82b446"), pt.pch = c(20, 18),
      xlab = "Relative time to treatment",
      main = "Event study: Staggered treatment (comparison)")

# True Effects
points(-20:20, true_effects, pch = 20, col = "grey60")

# Legend
legend(x=-20, y=3, col = c("steelblue", "#82b446", "grey60"), pch = c(20, 18, 20),
       legend = c("Two-stage estimate", "TWFE", "True Effect"))
```

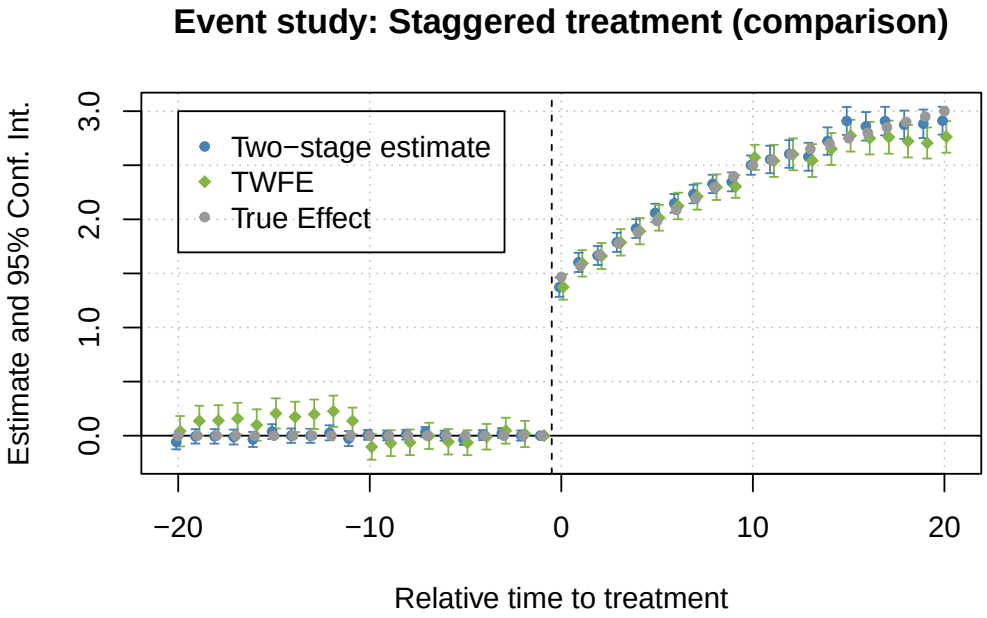


Figure 3: This figure adds the standard ordinary-least squares estimates to the true effect and the 'did2s' estimates present in Figure 2. The x-axis of this figure is the relative time to treatment, i.e. how many years pre-/post- treatment that period is. The y-axis is estimated treatment effects. There are three sets of points. The first two sets of points are the same as in Figure 2. The third set of points is the ordinary-least squares estimates. These points exhibit show evidence of parallel pre-trends failing.

The `event_study` and `plot_event_study` command

The command `event_study` presents a common syntax that estimates the event-study TWFE model for treatment-effect heterogeneity robust estimators recommended by the literature and returns all the estimates in a data.frame for easy plotting by the command `plot_event_study`. The general syntax is

```
event_study(
    data, yname, idname, tname, gname,
    estimator,
    xformla = NULL, horizon = NULL, weights = NULL
)
```

The option `data` specifies the data set that contains the variables for the analysis. The four other required options are all names of variables: `yname` corresponds with the outcome variable of interest; `idname` is the variable corresponding to the (unique) unit identifier, i; `tname` is the variable corresponding to the time period, t; and `gname` is a variable indicating the period when treatment first starts (group status).

Table 2: Event Study Estimators

Estimator and R package	Type	Comparison group	Main Assumptions
Gardner (2021) {did2s}	Imputes $Y(0)$	Not-yet- and/or Never-treated	• Parallel Trends for all units • Limited anticipation* • Correct specification of $Y(0)$
Borusyak, Jaravel, and Spiess (2021) {didimputation}	Imputes $Y(0)$	Not-yet- and/or Never-treated	• Parallel Trends for all units • Limited anticipation* • Correct specification of $Y(0)$
Callaway and Sant'Anna (2021) {did}	2×2 Aggregation	Either Not-yet- or Never-treated	• Parallel Trends for Not-yet-treated *or* Never-treated • Limited anticipation*
Sun and Abraham (2020) {fixest/sunab}	2×2 Aggregation	Not-yet- and/or Never-treated	• Parallel Trends for all units • Limited anticipation*
Roth and Sant'Anna (2021) {staggered}	2×2 Aggregation	Not-yet-treated	• Treatment timing is random • Limited anticipation*

This table summarizes the differences between various proposed event-study estimators in the econometric literature. It highlights the two different strategies that different estimators use, namely imputation and 2x2 aggregation. Importantly, it tries to show the differences in assumptions for each different estimator.

* Anticipation can be accounted for by adjusting 'initial treatment day' back x periods, where x is the number of periods before treatment that anticipation can occur.

ISSN 2073-4859

There are five main estimators available and the choice is specified for the `estimator` argument and are described in Table 2.[5] The following paragraphs will aim to highlight the differences and commonalities between estimators. These estimators fall into two broad categories. First, **did2s** and **didimputation** (Butts 2021) are `imputation-based` estimators as described above. Both rely on "residualizing" the outcome variable $\tilde{Y} = Y_{it} - \hat{\mu}_g - \hat{\eta}_t$ and then averaging those $\tilde{Y}$ to estimate the event-study average treatment effect τ^k. These two estimators return identical point estimates for post-treatment effects, but differ in their asymptotic regime and hence their standard errors.

The second type of estimator, which we label `2x2 aggregation`, takes a different approach for estimating event-study average treatment effects. The packages **did** (Callaway and Sant'Anna 2021), **fixest** and **staggered** (Roth and Sant'Anna 2021) first estimate τ_{gt} for all group-time pairs. To estimate a particular τ_{gt}, they use a two-period (periods t and $g-1$) and two-group (group g and a "control group") difference-in-differences estimator, known as a `2x2` difference-in-differences. The particular "control group" they use will differ based on estimator and is discussed in the next paragraph. Then, the estimator manually aggregate τ_{gt} across all groups that were treated for (at least) k periods to estimate the event-study average treatment effect τ^k.

These estimators do not all rely on the same underlying assumptions, so the rest of the table summarizes the primary differences between estimators. The comparison group column describes which units are utilized as comparison groups in the estimator and hence will determine which units need to satisfy a parallel trends assumption. For example, in some circumstances, treated units will look very different from never-treated units. In this case, parallel trends may only hold between units that receieve treatment at some point and hence only these units should be used in estimation. In other cases, for example if treatment is assigned randomly, then it's reasonable to assume that both not-yet- and never-treated units would all satisfy parallel trends.

For estimators labeled "Not-yet- and/or never-treated", the default is to use both not-yet- and never-treated units in the estimator. However, if all never-treated units are dropped from the data set before using the estimator, then these estimators will use only not-yet-treated groups as the comparison group. **did** provides an option to use either the not-yet- treated *or* the never- treated group as a comparison group depending on which group a researcher thinks will make a better comparison group. **staggered** will automatically drop units that are never treated from the sample and hence only use not-yet-treated groups as a comparison group.

The next column, `Main Assumptions`, summarize concisely the main theoretical assumptions underlying each estimator. First, the assumptions about parallel trends match the previous discussion on the correct comparison group. The only estimator that doesn't rely on a parallel trends assumption is **staggered** which relies on the assumption that *when* a unit receives treatment is random.

The next assumption, that is common across all estimators, is that there should be "limited anticipation" of treatment. In general, anticipatory effects are when units respond to treatment before it is *actually* implemented. For example, this can be common if the news of a policy triggers behavior responses before the treatment is put in place. "Limited anticipation" is when these anticipatory effects can only exist in a "few" pre-periods.[6] In any of these cases, "treatment" should be manually moved back by the maximum number of periods where anticipation can occur. For example, if treatment starts in 2012 and anticipatory effects are reasonably only possible 2 years before, this units' "group" should be labeled as 2010 in the data.

The `imputation-based` estimators require an additional assumption that the parametric model of $Y(0) = \mu_i + \eta_t + \varepsilon_{it}$ is correctly specified. This is because in the first stage, you have to accurately impute $Y(0)$ when residualizing Y which relies on the correct specification of $Y(0)$. The `2x2 aggregation` models do not estimate a parametric form of $Y(0)$ and hence only relies on a parallel trends assumption. While not in the table, it is worth noting that **did** allows for uniform inference of estimates. This addresses the problem that multiple hypotheses tests are being done by researchers (e.g. checking individually if all post period estimates are significant) by creating standard errors that adjust for multiple testing.

Example usage of `event_study` The result of `event_study` is a tibble in a `tidy` format (Robinson, Hayes, and Couch 2021) that contains point estimates and standard errors for each relative time indicator for each estimator. The results of `event_study` are stored as a dataframe with event-study term, the estimate, standard error, and a column containing which estimator is used for that estimate. This output dataframe will in turn be passed to `plot_event_study` for easy comparison. `plot_event_study` will return a `ggplot` object (Wickham 2016). We return to the `df_het` dataset to see example usage of these functions.

[5] Except for Sun and Abraham, the `estimator` option is the package name. For Sun and Abraham, the `estimator` option is `sunab`. A value of "all" will estimate all 5 estimators.

[6] There should be more periods before treatment in the sample than whatever number a "few" is.

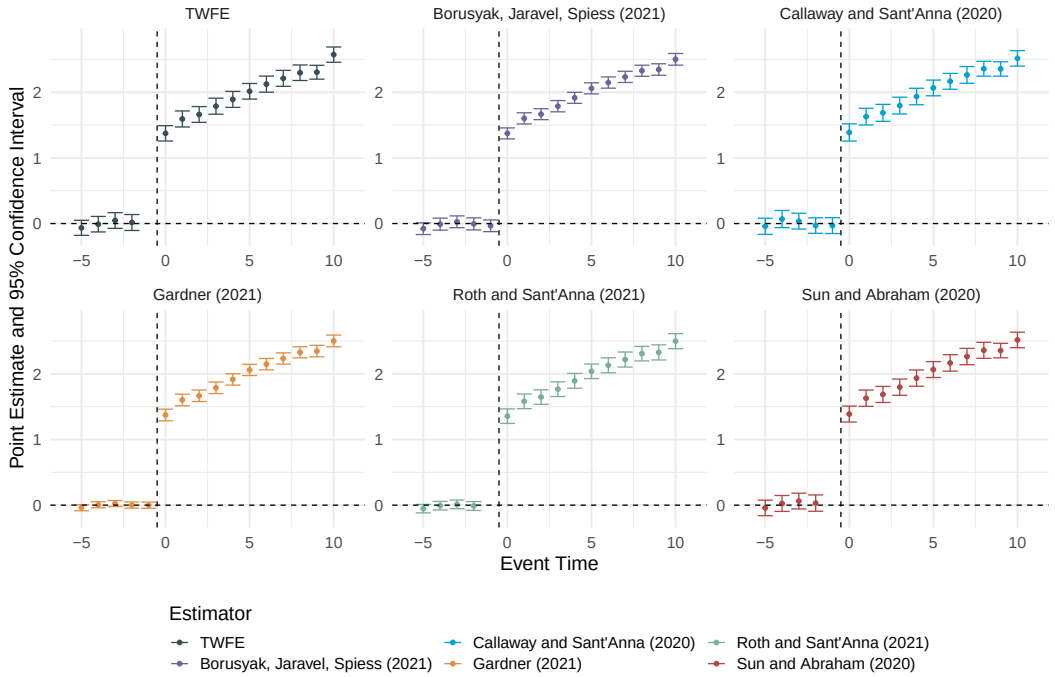


Figure 4: This figure contains six plots displayed in a grid of different event study estimators. The estimators are labeled 'TWFE', 'Borusyak, Jaravel, Spiess (2021)', 'Callaway and Sant'Anna (2020)', 'Gardner (2021)', 'Roth and Sant'Anna (2021)', and 'Sun and Abraham (2020)'. Each estimator's necessary assumptions are described above. Each plot in the figure displays point estimates from pre-treatment year -5 through post-treatment year 10. Each estimator is approximately 0 for all pre-treatment periods. In post-periods, each figure follows the true treatment effect starting at 1.5 in post-period 1 and growing afterwards.

```
data(df_het, package = "did2s")
out = event_study(
  data = df_het, yname = "dep_var", idname = "unit",
  tname = "year", gname = "g", estimator = "all"
)

head(out)

#>    estimator term   estimate  std.error
#> 1:      TWFE  -20 0.04097725 0.07167704
#> 2:      TWFE  -19 0.13665695 0.07147683
#> 3:      TWFE  -18 0.14015820 0.07245520
#> 4:      TWFE  -17 0.15793252 0.07431871
#> 5:      TWFE  -16 0.09910002 0.07379570
#> 6:      TWFE  -15 0.20561127 0.07116478

plot_event_study(out, horizon = c(-5, 10))
```

Conclusion

This article introduced the package **did2s** which provides a fast, memory-efficient, and treatment-effect heterogeneity robust way to estimate two-way fixed-effect models. The package also includes the `event_study` and `plot_event_study` functions to allow for a single syntax for the various estimators introduced in the literature. A companion package in Stata is also available with similar syntax for the `did2s` function.

While this package includes an event_study function that aims to help individuals implement any of the proposed modern "solutions" to the difference-in-differences estimation, further research on this topic is needed to help practitioners be able to more precisely determine which estimators work

best in their settings. Potentially, there could be data-driven methods to try to identify the plausibility of the different assumptions. Additionally, there is still more work to be done to formalize under what conditions covariates can flexibly be used in estimation. There is some initial work from Caetano et al. (2022), but there does not yet exist statistical software to perform their proposed estimator.

References

Bergé, Laurent. 2018. "Efficient Estimation of Maximum Likelihood Models with Multiple Fixed-Effects: The R Package FENmlm." *CREA Discussion Papers*, no. 13.

Borusyak, Kirill, Xavier Jaravel, and Jann Spiess. 2021. "Revisiting Event Study Designs: Robust and Efficient Estimation." Working Paper.

Butts, Kyle. 2021. *Didimputation: Difference-in-Differences Estimator from Borusyak, Jaravel, and Spiess (2021)*. https://www.github.com/kylebutts/didimputation.

Caetano, Carolina, Brantly Callaway, Stroud Payne, and Hugo Sant'Anna Rodrigues. 2022. "Difference in Differences with Time-Varying Covariates." Working Paper. http://arxiv.org/abs/2202.02903.

Callaway, Brantly, and Pedro H. C. Sant'Anna. 2020. "Difference-in-Differences with Multiple Time Periods." *Journal of Econometrics*. https://doi.org/https://doi.org/10.1016/j.jeconom.2020.12.001.

———. 2021. *Did: Difference in Differences*. https://bcallaway11.github.io/did/.

Chaisemartin, Clément de, and Xavier D'Haultfoeuille. 2019. *Two-Way Fixed Effects Estimators with Heterogeneous Treatment Effects*. w25904. National Bureau of Economic Research. https://doi.org/10.3386/w25904.

Frisch, Ragnar, and Frederick V. Waugh. 1933. "Partial Time Regressions as Compared with Individual Trends." *Econometrica* 1 (4): 387–401. http://www.jstor.org/stable/1907330.

Gardner, John. 2022. "Two-Stage Differences in Differences." arXiv. https://doi.org/10.48550/ARXIV.2207.05943.

Goodman-Bacon, Andrew. 2018. *Difference-in-Differences with Variation in Treatment Timing*. w25018. National Bureau of Economic Research. https://doi.org/10.3386/w25018.

Newey, Whitney, and Daniel McFadden. 1986. "Large Sample Estimation and Hypothesis Testing." In *Handbook of Econometrics*, edited by R. F. Engle and D. McFadden, 1st ed., 4:2111–2245. Elsevier. https://EconPapers.repec.org/RePEc:eee:ecochp:4-36.

Robinson, David, Alex Hayes, and Simon Couch. 2021. *Broom: Convert Statistical Objects into Tidy Tibbles*. https://CRAN.R-project.org/package=broom.

Roth, Jonathan, and Pedro H. C. Sant'Anna. 2021. *Staggered: Efficient Estimation Under Staggered Treatment Timing*. https://github.com/jonathandroth/staggered.

Sun, Liyang, and Sarah Abraham. 2020. "Estimating Dynamic Treatment Effects in Event Studies with Heterogeneous Treatment Effects." *Journal of Econometrics*, December. https://doi.org/10.1016/j.jeconom.2020.09.006.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Kyle Butts
University of Colorado Boulder

https://www.kylebutts.com/
ORCiD: 0000-0002-9048-8059
buttskyle96@gmail.com

John Gardner
University of Mississippi

https://jrgcmu.github.io/
jrgardne@olemiss.edu

rbw: An R Package for Constructing Residual Balancing Weights

by Derick S. Baum and Xiang Zhou

Abstract We describe the R package **rbw**, which implements the method of residual balancing weights (RBW) for estimating marginal structural models. In contrast to other methods such as inverse probability weighting (IPW) and covariate balancing propensity scores (CBPS), RBW involves modeling the conditional means of post-treatment confounders instead of the conditional distributions of the treatment to construct the weights. RBW is thus easier to use with continuous treatments, and the method is less susceptible to model misspecification issues that often arise when modeling the conditional distributions of treatments. RBW is also advantageous from a computational perspective. As its weighting procedure involves a convex optimization problem, RBW typically locates a solution considerably faster than other methods whose optimization relies on nonconvex loss functions — such as the recently proposed nonparametric version of CBPS. We explain the rationale behind RBW, describe the functions in **rbw**, and then use real-world data to illustrate their applications in three scenarios: effect estimation for point treatments, causal mediation analysis, and effect estimation for time-varying treatments with time-varying confounders.

1 Introduction

This paper describes the R package **rbw**, which implements the method of residual balancing weights for estimating marginal structural models (MSMs) (Zhou and Wodtke, 2020). MSMs seek to estimate causal effects in the presence of post-treatment confounding — a common issue in the social sciences. In studies of time-varying treatments, prior treatments may affect the confounders of future treatments. For example, research has shown that political candidates' decision to run negative advertisements is shaped by their positions in recent polling data, which are in turn affected by their previous decisions to run negative advertisements (Lau et al., 2007; Blackwell, 2013). Post-treatment confounding can also occur in causal mediation analysis when confounders of the mediator-outcome relationship are affected by the treatment. For example, such a problem arises in a study of the mediating role of morality in the effect of shared democracy on public support for war (Tomz and Weeks, 2013). Post-treatment variables, such as respondents' beliefs about the likelihood of victory, may affect both perceptions of morality and support for military intervention.

MSMs aim to address two types of bias associated with conventional regression methods that adjust naively for post-treatment confounders: overcontrol and collider-stratification bias (Robins, 1986; 2000). Conditioning naively on post-treatment confounders can create overcontrol bias as it blocks the effect of the treatment on the outcome that passes through these variables. Additionally, it can lead to collider-stratification bias when post-treatment confounders are affected by unobserved determinants of the outcome. This is because the adjustment will create a spurious association between the treatment and the unobserved variables.

Researchers often use inverse probability weighting (IPW) to fit MSMs (for an in-depth exposition of the method, see Robins et al., 2000; Robins, 2000; Cole and Hernán, 2008). In longitudinal settings, MSMs with IPW involve fitting a model for the conditional mean of the outcome given the treatment history using weights that break the dependence between past confounders and the treatment at each time point. In essence, the weights create a pseudo-population where the marginal mean of the potential outcomes under a treatment history equals the conditional mean of the observed outcome given the treatment history. The R package **ipw** provides functions for estimating inverse probability weights (van der Wal and Geskus, 2011; Geskus and van der Wal, 2015).

However, IPW's success depends on correct specification of the models for the conditional distributions of exposure to treatment and/or mediator (hereafter jointly referred to as "exposures"), which is difficult to achieve in practice. Moreover, even when these models are correctly specified, IPW is inefficient and susceptible to finite-sample bias (Zhou and Wodtke, 2020; Wang et al., 2006). Finally, when the exposures are continuous, IPW may perform poorly due to unreliable estimation of conditional densities (Naimi et al., 2014; Vansteelandt, 2009).

Alternative methods have attempted to mitigate these shortcomings. In particular, Imai and Ratkovic's (2014; 2015) covariate balancing propensity score (CBPS) method proposes a set of balancing conditions when estimating the propensity scores. Because it seeks to maximize the covariate balance between the treatment and control groups, this method is less sensitive to model misspecification than IPW. Fong et al. (2018) expand CBPS to accommodate continuous exposures, but the challenges involved in modeling conditional densities persist. With this in mind, they have also developed a

nonparametric extension that constructs weights that maximize the empirical likelihood while meeting a set of balancing conditions. Though the nonparametric CBPS (npCBPS) circumvents the need for specifying a functional form for the propensity score, it does so at a cost: since the empirical likelihood is not generally convex, the optimization procedure is often slow and may fail to find a solution. The latter can happen, for example, when we have a large number of covariates. The authors advance a workaround that adds flexibility to the covariate balancing conditions and penalizes the remaining imbalance. In doing so, they ensure that a weighting solution exists. Users can implement CBPS in R with the **CBPS** package (Fong et al., 2021).

Recently, Zhou and Wodtke (2020) propose the method of residual balancing weights (RBW) for estimating MSMs. RBW involves fitting models for the conditional means of post-treatment confounders given past treatments and confounders and extracting their residual terms. It then uses Hainmueller's (2012) entropy balancing method to find weights such that, in the weighted sample, 1) the residuals are orthogonal to future exposures, past treatments, and past confounders, and 2) the relative entropy between the weights and a set of base weights (e.g., survey sampling weights) is minimized. RBW is similar to npCBPS in that it relies on a set of balancing conditions to find the weights and does not require modeling the conditional distributions of the exposures. Both methods can, therefore, be easily adapted to cases where exposures are continuous.[1] Despite their similarities, RBW has a significant computational advantage: the relative entropy metric it uses to construct the weights leads to a convex optimization problem, so finding the weighting solution is computationally expeditious. As shown below, RBW manages to locate the solution considerably faster than npCBPS when we compare the methods' performance for the same problem.

In the sections that follow, we present an overview of the residual balancing method and how, in addition to contexts involving time-varying treatments, we can use it in cases of point treatments and causal mediation analysis. We then discuss the package that implements the method in R (**rbw**). Next, we describe the functions included in **rbw** and illustrate their use with various real-world data sets. The final section concludes.

2 Overview of residual balancing

This section gives an overview of RBW. We first describe the notation used throughout the paper and briefly review MSMs. Next, we explain the underlying logic of RBW and provide an intuition for how the method works using a directed acyclic graph (DAG).

Notation

Assume we have a study with $T \geq 2$ time points, and we are interested in the effect of a time-varying treatment, A_t ($1 \leq t \leq T$), on some end-of-study outcome Y. We also have a vector of observed time-varying confounders, L_t, at each time point, which may be affected by prior treatments. $\bar{A}_t = (A_1, ..., A_t)$ and $\bar{L}_t = (L_1, ..., L_t)$ denote treatment and covariate histories up to time t. Furthermore, $\bar{A} = \bar{A}_T$ and $\bar{L} = \bar{L}_T$ represent a respondent's complete treatment and covariate histories, respectively. Finally, let $Y(\bar{a})$ be the potential outcome under some treatment history $\bar{a}$.

MSMs

An MSM is a model for the marginal mean of the potential outcomes under some treatment history:

$$\mathbb{E}[Y(\bar{a})] = \mu(\bar{a}; \beta), \tag{1}$$

where $\mu(.)$ is some function and β are a set of parameters capturing the causal effects of interest. We can identify an MSM from observed data under three assumptions:

1. consistency: if $\bar{A} = \bar{a}$, then $Y = Y(\bar{a})$;
2. sequential ignorability: at each time point t, treatment is unconfounded conditional on past treatments and the covariate history up to that point. Formally, $Y(\bar{a}) \perp\!\!\!\perp A_t | \bar{A}_{t-1}, \bar{L}_t$; and

[1] Other methods for estimating causal effects of continuous exposures include doubly robust estimators, which model both the treatment and outcome processes and give consistent estimates as long as one of these models is correctly specified. For example, Kennedy et al. (2017) introduce an approach that does not require any parametric assumptions about the effect curve. Instead, it uses flexible data-adaptive methods both to estimate the treatment and outcome models and to subsequently fit the dose-response curve. Readers can install the R package that implements this method from GitHub (Kennedy, 2021).

3. positivity: at each time point t, treatment assignment must not be deterministic. That is, if $f(\bar{A}_{t-1} = \bar{a}_{t-1}, \bar{L}_t = \bar{l}_t) > 0$, then $f(A_t = a_t | \bar{A}_{t-1} = \bar{a}_{t-1}, \bar{L}_t = \bar{l}_t) > 0$, where $f(\cdot)$ represents a probability mass or density function.

Under these assumptions, Robins (1986) shows that the expected value of the potential outcome $\mathbb{E}[Y(\bar{a})]$ can be identified via the g-computation formula:

$$\mathbb{E}[Y(\bar{a})] = \int ... \int \mathbb{E}[Y|\bar{A} = \bar{a}, \bar{L} = \bar{l}] \prod_{t=1}^{T} f(l_t|\bar{l}_{t-1}, \bar{a}_{t-1}) d\mu(l_t), \tag{2}$$

where $\mu()$ is an appropriate dominating measure. While Equation 2 provides a general formula for identifying causal effects in the presence of time-varying treatments, directly evaluating it is often impractical, particularly when we have many covariates and time periods.

The rationale behind residual balancing

Now consider the formula for the conditional mean of the observed outcome Y given some treatment history:

$$\mathbb{E}[Y|\bar{A} = \bar{a}] = \int ... \int \mathbb{E}[Y|\bar{A} = \bar{a}, \bar{L} = \bar{l}] \prod_{t=1}^{T} f(l_t|\bar{l}_{t-1}, \bar{a}) d\mu(l_t). \tag{3}$$

By comparing Equations 2 and 3, we see that weighting the observed population by

$$W_l = \prod_{t=1}^{T} \frac{f(L_t|\bar{L}_{t-1}, \bar{A}_{t-1})}{f(L_t|\bar{L}_{t-1}, \bar{A})} \tag{4}$$

creates a pseudo-population in which $f^*(l_t|\bar{l}_{t-1}, \bar{a}) = f^*(l_t|\bar{l}_{t-1}, \bar{a}_{t-1}) = f(l_t|\bar{l}_{t-1}, \bar{a}_{t-1})$ and $\mathbb{E}^*[Y|\bar{A} = \bar{a}] = \mathbb{E}^*[Y(\bar{a})] = \mathbb{E}[Y(\bar{a})]$, where * represents quantities in the pseudo-population. Estimating the conditional densities of Equation 4 is challenging because L_t is often high-dimensional.

Zhou and Wodtke (2020) demonstrate that the condition $f^*(l_t|\bar{l}_{t-1}, \bar{a}) = f^*(l_t|\bar{l}_{t-1}, \bar{a}_{t-1}) = f(l_t|\bar{l}_{t-1}, \bar{a}_{t-1})$ implies a series of moment conditions in the pseudo-population. Most importantly,

$$\mathbb{E}^*[\delta(g(L_t))h(\bar{L}_{t-1}, \bar{A})] = \mathbb{E}^*[\delta(g(L_t))]\mathbb{E}^*[h(\bar{L}_{t-1}, \bar{A})] = 0, \tag{5}$$

where:

- $g(.)$ and $h(.)$ are scalar functions.
- $\delta(g(L_t))$ is the residual of $g(L_t)$ with respect to its conditional mean given the observed past: $\delta(g(L_t)) = g(L_t) - \mathbb{E}[g(L_t)|\bar{L}_{t-1}, \bar{A}_{t-1}]$.

Residual balancing aims to emulate the moment conditions (5) that would hold in the pseudo-population if it were possible to weight the observed population by W_l. To do so, the method 1) specifies a set of $g(\cdot)$ functions, $G(L_t) = \{g_1(L_t), ..., g_{J_t}(L_t)\}$ and a set of $h(.)$ functions, $H(\bar{L}_{t-1}, \bar{A}) = \{h_1(\bar{L}_{t-1}, \bar{A}), ..., h_{K_t}(\bar{L}_{t-1}, \bar{A})\}$; 2) computes a set of residual terms $\delta(g(L_t)) = g(L_t) - \mathbb{E}[g(L_t)|\bar{L}_{t-1}, \bar{A}_{t-1}]$; and 3) finds a set of weights such that, for any j, k, and t, the cross-moment of $\delta(g_j(l_{it}))$ and $h_k(\bar{l}_{i,t-1}, \bar{a}_i)$ is zero in the weighted data. That is, RBW locates the rbw_i weights subject to the following balancing conditions:

$$\sum_{i=1}^{n} rbw_i c_{ir} = 0, \;\; 1 \leq r \leq n_c, \tag{6}$$

where c_{ir} is the rth element of $\boldsymbol{c}_i = \{\delta(g_j(l_{it}))h_k(\bar{l}_{i,t-1}, \bar{a}_i); 1 \leq j \leq J_t, 1 \leq k \leq K_t, 1 \leq t \leq T\}$ and $n_c = \sum_{t=1}^{T} J_t K_t$ denotes the total number of balancing conditions. The residualized confounders at each time point are balanced across future treatments as well as past treatments and confounders (the observed past). RBW thus adjusts for post-treatment confounding without inducing overcontrol and collider-stratification bias.

Moreover, Zhou and Wodtke (2020) follow Hainmueller (2012) and minimize the relative entropy between rbw_i and a set of base weights q_i (e.g., survey sampling weights):

$$\min_{rbw_i} \sum_i rbw_i \log(rbw_i / q_i). \tag{7}$$

We can then use the method of Lagrange multipliers to find a weighting solution that minimizes the relative entropy between rbw_i and q_i subject to the n_c balancing constraints. We discuss this

procedure in greater depth below when describing the function `rbw::eb2()`. The convexity of the relative entropy metric renders it considerably more computationally efficient than nonconvex loss functions that can also be used to construct weights, such as the empirical likelihood (Fong et al., 2018).

A typical implementation of residual balancing can be summarized in three steps:

1. For each covariate j and at each time point t, fit a linear, logistic, or Poisson regression of l_{ijt} (depending on its level of measurement) on $\bar{l}_{i,t-1}$ and $\bar{a}_{i,t-1}$. Then compute the residuals $\hat{\delta}(l_{ijt})$. For the covariates in L_1 (the first time period), the residuals are computed as deviations from the sample mean: $\hat{\delta}(l_{ij1}) = l_{ij1} - \text{avg}(l_{j1})$. This step relies on the idea that $g_j(L_t) = L_{jt}$, where L_{jt} is the jth element of the covariate vector L_t, is a natural choice for the set of $g(.)$ functions constituting $G(L_t)$.
2. Find a set of weights, rbw_i, such that:
   a) in the weighted sample, the residuals $\hat{\delta}(l_{ijt})$ are orthogonal to all future treatments and the regressors of l_{ijt} (i.e., the past treatments and past confounders);
   b) the relative entroy between rbw_i and the base weights q_i is minimized.
3. Use the weights to fit an MSM.

Figure 1 depicts the logic of RBW in a DAG. Following the notation described above, A_t denotes our time-varying treatment, L_t is a vector of time-varying confounders, and Y is our end-of-study outcome. Further, assume two time points, $t = 1, 2$. The rbw_i weights break the dependence between A_t and $\bar{L}_t$ at each time point. That is, the weights create a pseudo-population where the confounding arrows $L_1 \to A_1$, $L_1 \to A_2$, and $L_2 \to A_2$ are broken while all others are preserved. It is important to note that L_1 is *marginally* independent of both A_1 and A_2 in the pseudo-population, but L_2 is *conditionally* independent of A_2, given L_1 and A_1. Hence, RBW invokes a model for the conditional mean of L_2 given L_1 and A_1 and balances the residuals from this model across levels of A_2 and levels of (L_1, A_1) (the observed past). This procedure avoids overcontrol and collider-stratification bias when adjusting for post-treatment confounding because it breaks the confounding arrow $L_2 \to A_2$ while leaving the causal arrow $A_1 \to L_2$ intact.

Finally, since $\mathbb{E}^*[Y|\bar{A} = \bar{a}] = \mathbb{E}^*[Y(\bar{a})] = \mathbb{E}[Y(\bar{a})]$ in the pseudo-population, we can estimate the marginal effects of interest by fitting a model for the conditional mean of the observed outcome given the treatment history (and possibly a set of baseline confounders) with weights equal to rbw_i.

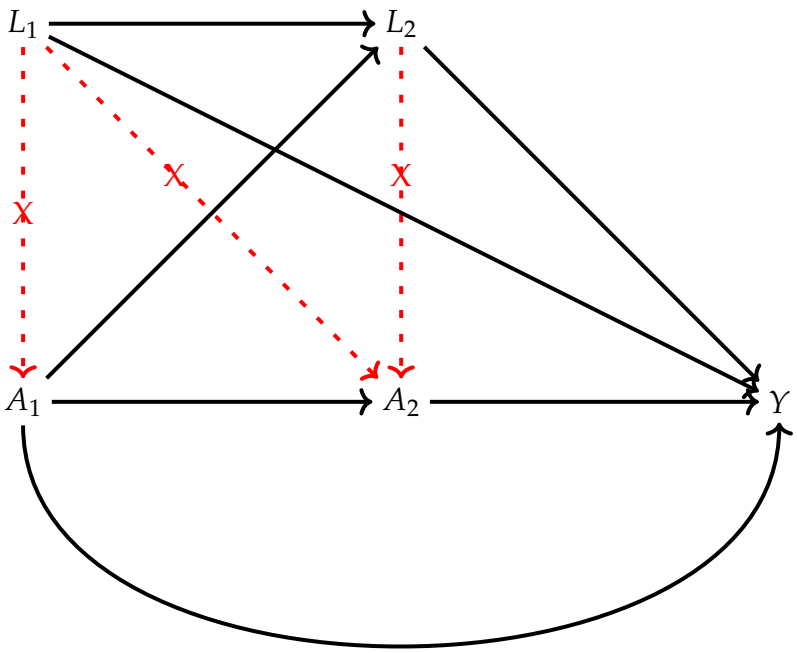


Figure 1: The underlying logic of residual balancing: A_t denotes the treatment at time t, L_t is a vector of time-varying confounders at time t, and Y is the end-of-study outcome. Residual balancing weights break the confounding arrows $L_1 \to A_1$, $L_1 \to A_2$, and $L_2 \to A_2$.

3 Uses of residual balancing

The rationale described in the previous section is targeted at estimating the causal effects of time-varying treatments. With minor adaptations, we can expand the use of RBW to two other contexts commonly encountered in the social and biomedical sciences: point treatment situations and causal mediation analysis.

RBW for estimating the average effect of a point treatment

RBW can be easily adapted to a point treatment situation where the user aims only to adjust for a set of baseline (i.e., time-invariant) confounders to estimate the average treatment effect. To this end, we modify the procedure above as follows:

1. Compute the response residuals $\hat{\delta}(x_{ij})$ for each baseline confounder X_j by centering it around its sample mean: $\hat{\delta}(x_{ij}) = x_{ij} - \text{avg}(x_j)$.
2. Find a set of weights, rbw_i, such that:
    a) in the weighted sample, the residuals $\hat{\delta}(x_{ij})$ are orthogonal to the treatment;
    b) the relative entropy between rbw_i and the base weights q_i is minimized.
3. Use the weights to fit an MSM.

The DAG of Figure 2 illustrates the point treatment situation. A represents the one-shot treatment, X is a vector of baseline confounders, and Y denotes our outcome. Weighting the observed population by rbw_i mimics a pseudo-population where the link between A and X is broken. We can then fit a model for the conditional mean of Y given A to estimate the causal effects of interest.[2]

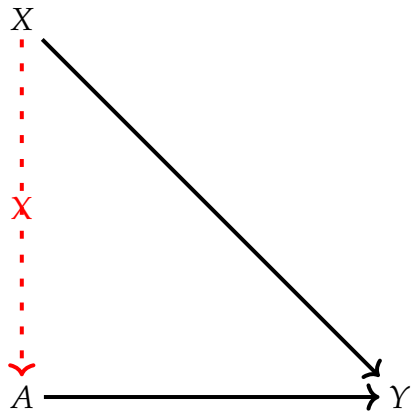


Figure 2: RBW in a point treatment: A represents the treatment, X is a vector of baseline confounders, and Y denotes the outcome. Residual balancing weights break the confounding arrow $X \to A$.

RBW in causal mediation analysis

In causal mediation analysis, researchers are often concerned with the joint effects of a one-shot treatment, A, and a mediator, M, on some end-of-study outcome Y when both baseline confounders (X) and some post-treatment confounders for the mediator-outcome relationship (Z) are present. With minor adjustments, the RBW implementation for causal mediation analysis is similar to the case with time-varying treatments:

1. As in the point treatment scenario, compute the response residuals $\hat{\delta}(x_{ij})$ for each baseline confounder X_j by centering it around its sample mean.
    - Note: users may skip this step by including the baseline confounders in the MSM in the final step.
2. Estimate the response residuals $\hat{\delta}(z_{ij})$ for each post-treatment confounder Z_j by fitting a linear, logistic, or Poisson regression of z_{ij} (depending on its level of measurement) on the treatment a_i and the baseline confounders x_i: $\hat{\delta}(z_{ij}) = z_{ij} - \mathbb{E}[z_{ij}|a_i, x_i]$.

[2]The vector X captures all confounding under the ignorability assumption.

3. Find a set of weights, rbw_i, such that:
    a) in the weighted sample, i) the baseline residuals $\hat{\delta}(x_{ij})$ are orthogonal to the treatment a_i and the mediator m_i; and ii) the post-treatment residuals $\hat{\delta}(z_{ij})$ are balanced across the treatment a_i, the mediator m_i, and the baseline confounders x_i;
    b) the relative entropy between rbw_i and the base weights q_i is minimized.
4. Use the weights to fit an MSM for the joint effects of the treatment and the mediator on the outcome:
    a) In causal mediation analysis, the potential outcomes of interest are denoted by $Y(a, m)$ (this is the potential outcome under treatment a and mediator value m). We can then express a saturated MSM as $\mathbb{E}[Y(a, m)] = \alpha_0 + \alpha_1 a + \alpha_2 m + \alpha_3 am$.
    b) Alternatively, the baseline confounders can be included in the MSM if users decide to skip the first step: $\mathbb{E}[Y(a, m)|X] = \alpha_0 + \alpha_1 a + \alpha_2 m + \alpha_3 am + \alpha_4^T X$.
    c) Finally, the controlled direct effects (CDE) of the treatment can be estimated as $\widehat{\text{CDE}}(m) = \mathbb{E}[Y(1, m) - Y(0, m)] = \hat{\alpha}_1 + \hat{\alpha}_3 m$. The CDE measures the causal effect of the treatment on the outcome when the mediator is fixed at some value m for all units.

4 The R package

The R package **rbw** contains four functions:

- `eb2()`, for generating minimum entropy weights subject to a set of balancing constraints.
- `rbwPoint()`, for constructing residual balancing weights to estimate the causal effects of one-shot treatments.
- `rbwMed()`, for constructing residual balancing weights to estimate controlled direct effects in causal mediation analysis.
- `rbwPanel()`, for constructing residual balancing weights to estimate the marginal effects of time-varying treatments.

Next, we explain each of these functions. The package also includes several real-world data sets (`advertisement`, `peace`, `campaign_long`, and `campaign_wide`), which we describe and analyze in the examples below.

Function `eb2()`

This function is an adaptation of `ebal::eb()` (Hainmueller, 2014). It is called internally by other functions in **rbw** to implement the method of Lagrange multipliers for locating a weighting solution that minimizes the relative entropy between rbw_i and q_i subject to the set of n_c balancing constraints described in Equation 6. Zhou and Wodtke (2020) impose an additional normalization constraint that ensures that the residual balancing weights sum to the sample size: $\sum_i rbw_i = n$. Following Hainmueller (2012), the authors obtain the primal optimization problem:

$$\min_{rbw_i} L^p = \sum_{i=1}^{n} rbw_i \log(rbw_i/q_i) + \sum_{r=1}^{n_c} \lambda_r \sum_{i=1}^{n} rbw_i c_{ir} + \lambda_0 \left(\sum_{i=1}^{n} rbw_i - n \right), \tag{8}$$

where $\{\lambda_1, ..., \lambda_{n_c}\}$ are the Lagrange multipliers for the balancing constraints and λ_0 is the Lagrange multiplier for the normalization constraint. Since the loss function L^p is strictly convex, the first order condition of $\frac{\partial L^p}{\partial rbw_i} = 0$ implies that the solution for each weight is

$$rbw_i^* = \frac{nq_i \exp(-\sum_{r=1}^{n_c} \lambda_r c_{ir})}{\sum_{i=1}^{n} q_i \exp(-\sum_{r=1}^{n_c} \lambda_r c_{ir})}. \tag{9}$$

We can then insert Equation 9 into L^p, leading to an unrestricted dual problem:

$$\max_{\lambda_r} L^d = -\log \left(\sum_{i=1}^{n} q_i \exp \left(-\sum_{r=1}^{n_c} \lambda_r c_{ir} \right) \right), \tag{10}$$

or equivalently,

$$\min_{Z} L^d = \log \left(Q' \exp(CZ) \right), \tag{11}$$

where $Q = [q_1, ..., q_n]'$, $C = [\boldsymbol{c}_1, ..., \boldsymbol{c}_1]'$, and $Z = -[\lambda_1, ..., \lambda_{n_c}]'$. Since L^d is strictly convex, the solution is guaranteed to be unique — assuming one exists. Given that both the gradient and the Hessian have closed-form expressions, we can solve the problem using Newton's method. `eb2()` implements the algorithm. If convergence is successful, the function tells the user that "Entropy minimization converged within tolerance level." Otherwise, it warns that entropy minimization did not converge and suggests increasing the number of iterations or reducing the number of balancing constraints.

The convexity of our optimization problem leads to appreciable computational gains over other methods that use alternative loss functions. This will be demonstrated later when we compare the performance of RBW with that of npCBPS — which uses the empirical likelihood — for the same problem.

`eb2()` is used as:

```
eb2(C, M, Q, Z = rep(0, ncol(C)), max_iter = 200, tol = 1e-04, print_level = 1)
```

and takes the following arguments:

- `C` is a constraint matrix, with each column corresponding to a balancing constraint.
- `M` is a vector of moment conditions to be met in the reweighted sample (per Equation 6, this is a vector of zeros with length equal to the number of columns of `C` when the other functions in **rbw** call `eb2()` internally).
- `Q` is a vector of base weights.
- `Z` is a vector of Lagrange multipliers to be initialized.
- `max_iter` determines the maximum number of iterations for Newton's method.
- `tol` is a tolerance parameter used to determine convergence. Specifically, convergence is achieved if `tol` is greater than the maximum absolute value of the deviations between the moments of the reweighted data and the target moments (i.e., `M`).
- `print_level` determines the level of printing:
  - **1** normal: print whether the algorithm converges or not.
  - **2** detailed: print also the maximum absolute value of the deviations between the moments of the reweighted data and the target moments in each iteration.
  - **3** very detailed: print also the step length of the line searcher in iterations where a full Newton step is excessive.

The output returned by `eb2()` is a list containing the following elements:

- `W` is a vector of normalized minimum entropy weights.
- `Z` is a vector of Lagrange multipliers.
- `converged` is a logical indicator for convergence.
- `maxdiff` is a scalar indicating the maximum absolute value of the deviation between the moments of the reweighted data and the target moments.

Function `rbwPoint()`

This function produces residual balancing weights to be used in a point treatment situation. It first takes a set of baseline confounders and computes the residuals for each confounder by centering it around its sample mean. Then it calls `eb2()` to find a set of weights, rbw_i, such that 1) the baseline residuals are orthogonal to the treatment in the weighted sample, and 2) the relative entropy between rbw_i and the base weights is minimized. Additionally, `rbwPoint()` calls a function that ensures that the matrix of balancing constraints comprises only linearly independent columns.

`rbwPoint()` is used as:

```
rbwPoint(treatment, data, baseline_x, base_weights, max_iter = 200,
    tol = 1e-04, print_level = 1)
```

and takes the following arguments:

- `max_iter`, `print_level`, and `tol` have the same definitions as in `eb2()`.
- `treatment` is a symbol or character string for the treatment variable.
- `data` is a data frame containing all variables in the model.
- `baseline_x` is an expression for a set of baseline confounders stored in `data` or a character vector of the names of these variables.
- `base_weights` is an *optional* vector of base weights (or its name). If no value is supplied, the function sets a vectors of ones with length equal to the sample size as the base weights.

The output returned by `rbwPoint()` is a list containing the following elements:

- `weights` is a vector of residual balancing weights.

- `constraints` is a matrix of linearly independent residual balancing constraints.
- `eb_out` contains the results from calling the `eb2()` function.
- `call` is the matched call (the function call with all arguments specified by their full names).

Function `rbwMed()`

This function produces residual balancing weights for causal mediation analysis. It takes an *optional* set of baseline confounders (as explained above, users can opt instead to include these covariates in the MSM later) and a list of model objects for the conditional mean of each post-treatment confounder given the treatment and baseline confounders. It then calls `eb2()` to find a set of weights, rbw_i, such that, in the weighted sample, 1) the baseline residuals are orthogonal to the treatment and the mediator; 2) the post-treatment confounders are balanced across the treatment, the mediator, and the baseline confounders; and 3) the relative entropy between rbw_i and the base weights is minimized.

`rbwMed()` takes an additional argument, `interact`, a logical variable indicating whether the baseline and post-treatment confounders should be balanced against the treatment-mediator interaction. This argument is set to `FALSE` by default, but users suspecting an interaction effect may find it prudent to balance against it. Like `rbwPoint()`, `rbwMed()` calls a function internally to ensure that only linearly independent columns constitute the matrix of balancing constraints.

It is used as:

```
rbwMed(treatment, mediator, zmodels, data, baseline_x, interact = FALSE,
    base_weights, max_iter = 200, tol = 1e-04, print_level = 1)
```

and takes the following arguments:

- `treatment`, `data`, `base_weights`, `max_iter`, `print_level`, and `tol` are defined as in `rbwPoint()`.
- `baseline_x` is an *optional* expression for a set of baseline confounders stored in `data` or a character vector of the names of these variables.
- `mediator` is a symbol or character string representing the mediator variable.
- `zmodels` is a list of fitted `lm` or `glm` objects for post-treatment confounders of the mediator-outcome relationship. Users should set this argument to `NULL` if there are no post-treatment confounders.
- `interact` is a logical variable indicating whether baseline and post-treatment confounders should be balanced against the treatment-mediator interaction term(s).

The output returned by `rbwMed()` is a list containing the same elements as the output from `rbwPoint()`.

Function `rbwPanel()`

This function produces residual balancing weights for estimating the marginal effects of time-varying treatments. It takes a list of model objects for the conditional mean of each post-treatment confounder given past treatments and past confounders. Then it calls `eb2()` to find a set of weights, rbw_i, such that 1) residuals of the post-treatment confounders are orthogonal to future treatments and the observed past in the weighted sample, and 2) the relative entropy between rbw_i and the base weights is minimized. Like the other functions, `rbwPanel()` ensures that the matrix of balancing constraints consists only of linearly independent columns.

It is used as:

```
rbwPanel(treatment, xmodels, id, time, data, base_weights, future = 1L,
    max_iter = 200, tol = 1e-04, print_level = 1)
```

and takes the following arguments:

- `data`, `base_weights`, `max_iter`, `print_level`, and `tol` are defined as in `rbwPoint()` and `rbwMed()`.
- `treatment` is a symbol or character string for the time-varying treatment.
- `xmodels` is a list of fitted `lm` or `glm` objects for time-varying confounders.
- `id` is a symbol or character string for the unit id variable.
- `time` is a symbol or character string for the time variable.
- `future` is an integer indicating the number of future treatments in the balancing conditions. When `future >0`, the residualized time-varying covariates are balanced not only with respect to current treatment A_t, but also with respect to future treatments $A_{t+1}, ..., A_{t+\texttt{future}}$. The default, `future = 1`, assumes away higher-ordered lagged effects of the covariates on the treatment. Users can leave out lagged effects entirely by setting `future` to zero.

The output returned by `rbwPanel()` is essentially the same as those from `rbwPoint()` and `rbwMed()`. The only difference is that the `weights` object, instead of a vector, is now a data frame with two columns: the id variable and the residual balancing weights (the column storing the weights is called `rbw`).

5 Examples

We now illustrate the functions `rbwPoint()`, `rbwMed()`, and `rbwPanel()` with data sets `advertisement`, `peace`, `campaign_long`, and `campaign_wide`, which are included in **rbw**. We expect users to rarely need to call `eb2()` manually since its primary use is to be called internally by the other functions. Hence, we see little gain in providing a separate example for it.

Point treatment: effects of political advertisements on campaign contributions

Urban and Niebler (2014) studied the effects of televised political advertisements on campaign contributions. Presidential candidates do not tend to deliberately advertise in states where competition for electoral votes is tame. Yet, some areas of noncompetitive states have overlapping media markets with battleground states. Because these media market spillovers do not encompass other forms of campaigning (e.g., rallies, speeches, etc.), the authors can isolate the effect of television advertising by restricting their analyses to noncompetitive states. Their original method involved estimating the propensity score with a logistic model and then conducting propensity score matching. To do so, they dichotomized the political advertising variable to indicate whether a zip code received more than 1000 advertisements.

Deeming this approach inadequate — in part because balancing against a dichotomous treatment does not ensure covariate balance on the underlying continuous variable — Fong et al. (2018) revisit the study using the CBPS method applied to a continuous treatment. Next, we examine how RBW fares compared with CBPS and IPW in this point treatment situation.

Importantly, CBPS assumes that the treatment variable is normally distributed. To satisfy this assumption, Fong et al. (2018) search across Box-Cox transformations to find the most appropriate transformation of the treatment. Since Q-Q plots show no sizable differences between the Box-Cox approach and a simple log transformation in achieving normality, we favor the latter to avoid extraneous details that would divert the example from its primary objective of illustrating RBW's implementation. Hence, we have $a_i^* = \log(a_i + 1)$, where a_i is the original treatment variable, the total number of political advertisements in a zip code, and a_i^* is the transformed treatment (we add one to a_i inside the log to avoid $\log(0)$ for zip codes receiving no advertisements). We also have a vector of baseline confounders consisting of the zip code's log population, population density, log median income, percent Hispanic, percent black, percent over age 65, percent college graduates, and a binary indicator of whether it is possible to commute to the zip code from a competitive state. Finally, Y represents the outcome, campaign contributions in thousands of dollars.

Our MSM includes the transformed treatment variable and state dummies U to account for state fixed effects:

$$\mathbb{E}[Y(a^*)|U] = \theta_0 + \theta_1 a^* + \theta_2^T U. \tag{12}$$

The data set `advertisement` contains the variables necessary for the analyses. We start by loading the necessary packages and our data set:

```
library("rbw")
data("advertisement")
```

`rbwPoint()` will construct the residual balancing weights by following the steps described above. It first computes the baseline residuals $\hat{\delta}(x_{ij}) = x_{ij} - \text{avg}(x_j)$ and then finds a set of weights such that, in the weighted sample, $\hat{\delta}(x_{ij})$ are orthogonal to the treatment, and the relative entropy between the residual balancing weights and a set of base weights is minimized. Since `advertisement` does not include sampling weights, `rbwPoint()` will set a vector of ones with length equal to the sample size as the base weights.

```
rbwPoint_fit <-
  rbwPoint(
    treatment = treat,
    baseline_x = c(
      log_TotalPop,
      PercentOver65,
```

```
      log_Inc,
      PercentHispanic,
      PercentBlack,
      density,
      per_collegegrads,
      CanCommute
    ),
    data =
      advertisement
  )
```

Next, we attach the rbw_i weights to the data:

```
advertisement$rbwPoint_weights <- rbwPoint_fit$weights
```

Following most applications of MSMs, we compute standard errors using the robust ("sandwich") variance estimator. This can be implemented with the function `survey::svydesign()`, which allows us to specify a complex survey design and estimate standard errors consistent with this specification.[3]

```
library("survey")
rbwPoint_design <- svydesign(ids = ~ 1,
                             weights = ~ rbwPoint_weights,
                             data = advertisement)
```

We then use the residual balancing weights to fit the MSM defined in Equation 12:

```
rbwPoint_msm <- svyglm(Cont ~ treat + factor(StFIPS),
                       design = rbwPoint_design)
```

Since we have transformed our treatment variable, we need to make the necessary adjustments to compute the treatment effect. As Urban and Niebler (2014) suggest, it is informative to study the effect of going from zero to 1,000 political advertisements on campaign contributions. Hence, we create a `dose` variable to account for this. If $a_i^* = \log(a_i + 1)$, we can define our dose as $\log(1000 + 1)$:

```
dose <- log(1000 + 1)
```

To find the estimate $\hat{\tau}_{rbw}$ of the average treatment effect, we multiply the coefficient for the transformed treatment variable by our dose. We also multiply it by 1,000 since the outcome variable is measured in thousands of dollars:

```
rbwPoint_tau <- 1000 * coef(rbwPoint_msm)[2] * dose
```

Next, we use the basic properties of the variance operator to derive the standard error. Again, we multiply the result by 1,000 given the scale of the outcome variable:

```
rbwPoint_vcov <- stats::vcov(rbwPoint_msm)
rbwPoint_se <- 1000 * dose * sqrt(rbwPoint_vcov[2, 2])
```

We also report the results using the functions from the **CBPS** and **ipw** packages. Recall from the introduction that the parametric CBPS is similar to IPW in that it requires explicit models for the conditional distributions of the treatment. However, it improves IPW by considering a set of balancing conditions during the propensity score estimation, thereby reducing sensitivity to model misspecification. By contrast, the nonparametric CBPS (npCBPS) does not require direct estimation of the propensity score; instead, it finds weights that maximize the empirical likelihood while meeting a set of balancing constraints. As such, like RBW, it avoids the need to specify a propensity score model. Readers can find the code detailing the construction of the CBPS and IPW weights in the supplementary material.

Table 1 summarizes the findings. All methods yield relatively similar point estimates and standard errors. In particular, they indicate that going from zero to 1,000 political advertisements increases campaign contributions by around four thousand dollars, on average, although the point estimates from CBPS and IPW are slightly larger than those produced by RBW and npCBPS. The last column shows that different loss functions for the optimization problem can lead to stark differences in computation time. While RBW's relative entropy metric leads to a convex optimization problem that Newton's method can solve in less than one second, npCBPS's algorithm takes much longer to run.

[3] Zhou and Wodtke (2020) conduct a set of simulation studies to assess the performance of the robust variance estimator across different methods. They find that the estimator tends to overestimate the true sampling variance for residual balancing across nearly all scenarios, making it consistently conservative for RBW. By contrast, the estimator sometimes overestimates and other times underestimates the true sampling variance for other weighting methods, including CBPS.

Table 1: Comparison of RBW, npCBPS, CBPS, and IPW for a Point Treatment Situation

Method	Estimate	Standard Error	Computation Time (in Seconds)
RBW	4043	2131	0.51
npCBPS	3914	2091	118.70
CBPS	4181	2118	7.71
IPW	4118	2078	0.04

Computation time may differ depending on system setup.
System setup used to generate the results:
MacBook Pro (15-inch, 2018), 2.2 GHz 6-Core Intel Core i7, 16GB RAM.

We next illustrate the use of `rbwMed()` and `rbwPanel()`.

Causal mediation analysis: the controlled direct effect of shared democracy on public support for war

A stylized fact in political science is that democracies do not engage in war with one another. To assess the role of public opinion in keeping the peace, Tomz and Weeks (2013) designed survey experiments that presented participants with a situation where a country was developing nuclear weapons. When describing the situation, the authors randomly and independently changed three characteristics of the country: its political regime (whether it was a democracy), alliance status (whether it had signed a military alliance with the United States), and economic ties (whether it had high levels of trade with the US). The outcome of interest was the respondent's support for military action on a five-point scale ranging from "oppose strongly" to "favor strongly." The authors found that respondents were considerably less supportive of military action against democracies than otherwise identical autocratic regimes.

Tomz and Weeks (2013) then went on to investigate the causal mechanisms through which shared democracy reduces public enthusiasm for war. In particular, they measured individuals' beliefs about the level of threat posed by the potential adversary (number of adverse events respondents considered probable if the US did not engage in war); the cost of the intervention (number of negative consequences anticipated if the US engaged in war); and the likelihood of success (a three-point scale assigning values of 0, 1, and 2 to the following beliefs, respectively: the operation had less than a 50-50 chance of working even in the short run, it would succeed only in the short run, and it would be successful both in the short and long run). They also collected data on people's moral concerns about using military force.

Their methodological framework focused on estimating the natural direct and natural indirect effects (Imai et al., 2011, 2010), whose identification assumptions require that no post-treatment confounding of the mediator-outcome relationship exists. As such, the authors examined the role of each mediator separately by assuming they operate independently of one another. Yet, as discussed in Zhou and Wodtke (2020), beliefs about the threat, cost, and likelihood of success may affect an individual's perceptions of morality while also influencing support for war directly. By treating these variables as post-treatment confounders, Zhou and Wodtke (2020) analyzed the mediating role of morality using controlled direct effects.

Let A denote the treatment, whether the country developing nuclear weapons was presented as a democracy, M the mediator, a dummy variable indicating whether the participant deemed the military action morally wrong, and Y the outcome, the respondent's support for war on a five-point scale. We also have a set of baseline confounders (X) including dummies for the other two randomized treatments (alliance status and economic ties) in addition to several demographic and attitudinal controls.[4] Finally, Z is the vector of post-treatment confounders comprising measures of beliefs about the threat, cost, and likelihood of success.

Our MSM is thus defined as:

$$\mathbb{E}[Y(a,m)] = \alpha_0 + \alpha_1 a + \alpha_2 m + \alpha_3 am. \tag{13}$$

We can alternatively include the baseline confounders in the MSM instead of balancing them across the treatment and the mediator with baseline residuals:

[4]For example, attitudinal controls include respondents' attitudes toward ethnocentrism measured with a series of questions about their opinions on immigration, affirmative action, and gay marriage.

$$\mathbb{E}[Y(a,m)|X] = \alpha_0 + \alpha_1 a + \alpha_2 m + \alpha_3 am + \alpha_4^T X. \tag{14}$$

The peace data set includes the variables we will use in the analyses. Let us first consider the approach where the baseline confounders are balanced across the treatment and the mediator using the baseline residuals computed from centering each covariate around its sample mean.

As explained above, RBW in causal mediation analysis requires models for the conditional means of each post-treatment confounder Z_j given the treatment a_i and the baseline confounders x_i: $\hat{\mathbb{E}}[z_{ij}|a_i, x_i]$. Hence, we assume that `threatc`, `cost`, and `successc` are measured on a continuous scale and fit linear models for each:

```
data("peace")
z1 <- lm(threatc ~ ally + trade + h1 + i1 + p1 + e1 + r1 +
           male + white + age + ed4 + democ,
         data = peace)
z2 <- lm(cost ~ ally + trade + h1 + i1 + p1 + e1 + r1 +
           male + white + age + ed4 + democ,
         data = peace)
z3 <- lm(successc ~ ally + trade + h1 + i1 + p1 + e1 + r1 +
           male + white + age + ed4 + democ,
         data = peace)
```

We then store the three model objects together in a list to be passed later to the `zmodels` argument in `rbwMed()`:

```
zmodels <- list(z1, z2, z3)
```

To construct the residual balancing weights, `rbwMed()` will 1) compute the baseline residuals $\hat{\delta}(x_{ij}) = x_{ij} - \text{avg}(x_j)$ and the post-treatment residuals $\hat{\delta}(z_{ij}) = z_{ij} - \mathbb{E}[z_{ij}|a_i, x_i]$ and 2) find a set of weights such that a) in the weighted sample, the baseline and post-treatment residuals meet the orthogonality requirements described earlier, and b) the relative entropy between the residual balancing weights and a set of base weights is minimized. The function will use a vector of ones as the base weights since `peace` does not include sampling weights. We also pass the name of our mediator to the `mediator` argument:

```
rbwMed_fit <- rbwMed(
  treatment = democ,
  mediator = immoral,
  zmodels = zmodels,
  baseline_x = c(ally, trade, h1, i1,
                 p1, e1, r1, male, white, age, ed4),
  data = peace
)

#> Entropy minimization converged within tolerance level
```

Next, we attach the weights to `peace`:

```
peace$rbwMed_weights <- rbwMed_fit$weights
```

We use `survey::svydesign()` to estimate robust standard errors:

```
rbwMed_design <- svydesign(ids = ~ 1,
                           weights = ~ rbwMed_weights,
                           data = peace)
```

Finally, we use the residual balancing weights to fit the MSM from Equation 13:

```
rbwMed_msm <- svyglm(strike ~ democ * immoral,
                     design = rbwMed_design)
```

Steps are similar for the case where the baseline confounders are not balanced against the treatment and the mediator but instead adjusted in the MSM, as defined in Equation 14. The models for the conditional means of the post-treatment confounders are the same as before, but we now leave the `baseline_x` argument empty:

```
rbwMed2_fit <- rbwMed(
  treatment = democ,
  mediator = immoral,
  zmodels = zmodels,
  data = peace
)

#> Entropy minimization converged within tolerance level

peace$rbwMed2_weights <- rbwMed2_fit$weights
rbwMed2_design <- svydesign(ids = ~ 1,
                            weights = ~ rbwMed2_weights,
                            data = peace)
rbwMed2_msm <- svyglm(strike ~ ally + trade + h1 + i1 + p1 +
                        e1 + r1 + male + white + age + ed4 + democ * immoral,
                      design = rbwMed2_design)
```

We summarize the results in Table 2. The estimated CDE if respondents lacked any moral qualms about military intervention, i.e., $\widehat{\text{CDE}}(m=0) = \hat{\mathbb{E}}[Y(1,0) - Y(0,0)] = \hat{\alpha}_1$, is -0.32 under the first approach and -0.36 under the second. Both estimates are statistically significant at the level of 0.01. The estimated CDE if respondents had moral qualms about military intervention, i.e., $\widehat{\text{CDE}}(m=1) = \hat{\mathbb{E}}[Y(1,1) - Y(0,1)] = \hat{\alpha}_1 + \hat{\alpha}_3$, is -0.36 under the first approach and -0.22 under the second.

Table 2: MSM Results for the Controlled Direct Effects of Shared Democracy on Public Support for War: $\widehat{\text{CDE}}(m=0)$ Equals the Coefficient for Shared Democracy; $\widehat{\text{CDE}}(m=1)$ Equals the Coefficient for Shared Democracy Plus the Coefficient for the Interaction Term

	Baseline Confounders Balanced	Baseline Confounders Adjusted in the MSM
Shared democracy	-0.317^{***}	-0.360^{***}
	(0.098)	(0.080)
Moral concerns	-1.272^{***}	-1.201^{***}
	(0.175)	(0.135)
Shared democracy * Moral concerns	-0.048	0.138
	(0.210)	(0.157)

*p<0.1; **p<0.05; ***p<0.01
Robust standard errors in parentheses.

Time-varying treatment: the cumulative effect of negative advertising on vote shares

We conclude this section with an example of RBW applied to a context involving time-varying treatments and confounders. Political scientists have shown interest in examining the cumulative effect of negative campaign advertising on election outcomes (Lau et al., 2007; Blackwell, 2013; Imai and Ratkovic, 2015). This is an intricate process because while polling results affect current campaign strategies, they are also constantly shifting, as they respond both to previous results and candidates' use of negative advertising in the past. For their ability to accommodate dynamic causal relationships — particularly since they allow past treatments to affect current outcomes (i.e., "carryover effects") and past outcomes to influence current treatment (i.e., "feedback effects") (Imai and Kim, 2019) — MSMs are suitable for this research question.

We use a dataset on the campaign of 113 Democratic candidates in US Senate and Gubernatorial Elections from 2000 to 2006. Let A_t represent our continuous treatment, the proportion of campaign advertisements mentioning the adversary in each campaign week, L_t our time-varying confounders, the Democratic share and the share of undecided voters in the polls, and Y the outcome, the Democratic share of the two-party vote. Additionally, we have a set of baseline confounders X including total campaign length, election year, and whether the election is senatorial or gubernatorial.

Zhou and Wodtke (2020) define an MSM as follows:

$$\mathbb{E}[Y(\bar{a})|X] = \beta_0 + \beta_1 \text{avg}(\bar{a}) + \beta_2 V + \beta_3 \text{avg}(\bar{a}) V + \beta_4^T X, \tag{15}$$

where V is an indicator of incumbency status used to construct interactions that allow the effect to differ between incumbents and nonincumbents, and $\text{avg}(\bar{a})$ is the average proportion of advertisements that were negative over the final five weeks of the campaign multiplied by ten (following Zhou and Wodtke (2020), we multiply this quantity by ten so that the regression coefficients can be interpreted as the effect of a ten-percentage point increase in negative advertising).

rbw contains two data sets associated with this problem: `campaign_long` and `campaign_wide`. They represent, respectively, the long-format and wide-format data on negative campaign advertising.

Recall that RBW requires us to fit a model for the conditional mean of each covariate at each time point given the observed past. We thus estimate regression models of our time-varying confounders L_t ($t \geq 2$) on lagged treatment and lagged confounders. We also interact each regressor with the week dummies, thus allowing the coefficients to change over time in a flexible manner:

```
data("campaign_long")
data("campaign_wide")
x1 <-
  lm(dem.polls ~ (neg.dem.l1 + dem.polls.l1 + undother.l1) * factor(week),
     data = campaign_long)
x2 <-
  lm(undother ~ (neg.dem.l1 + dem.polls.l1 + undother.l1) * factor(week),
     data = campaign_long)
```

We then create a list with the model objects to be passed later to the `xmodels` argument in `rbwPanel()`:

```
xmodels <- list(x1, x2)
```

To construct the residual balancing weights, `rbwPanel()` first extracts the residual terms $\hat{\delta}(L_t)$ from the models above. Note that for each covariate in L_1 (the first time period), the residuals are computed as deviations from the sample mean. Then the function finds a set of weights, such that, in the weighted sample, the residuals are orthogonal to current and future treatments as well as the regressors of L_{jt}, and the relative entropy between the residual balancing weights and the base weights is minimized. Note that we set the `future` argument to zero to replicate the results from Zhou and Wodtke (2020) since the authors balance the residualized time-varying confounders only with respect to the current treatment, thus assuming away lagged effects of the covariates on the treatment.[5] Since our data do not include sampling weights, a vector of ones is used as the base weights. Additionally, we need to pass arguments indicating the unit id and the time variables due to the longitudinal structure of our data set.

```
rbwPanel_fit <- rbwPanel(
  treatment = neg.dem,
  xmodels = xmodels,
  id = id,
  time = week,
  data = campaign_long,
  future = 0
)

#> Entropy minimization converged within tolerance level
```

We now attach the weights to `campaign_wide` using the pipe operator and the `left_join()` function from the package **dplyr** (this merging is permitted because the `weights` object returned by `rbwPanel()` is a data frame containing the id variable and the residual balancing weights):

```
library("dplyr")
campaign_wide <- campaign_wide %>%
  left_join(rbwPanel_fit$weights, by = "id")
```

We use the functions from **dplyr** to rename the weights so that our variable's name is consistent with the previous examples:

```
campaign_wide <- campaign_wide %>%
  rename(rbwPanel_weights = rbw)
```

Again, we use `survey::svydesign()` to compute robust standard errors:

[5] The negative advertising data set spans five weeks, so lagged effects of the time-varying covariates on the treatment can be incorporated by setting `future > 0`, up to `future = 4`.

```
rbwPanel_design <- svydesign(ids = ~ 1,
                             weights = ~ rbwPanel_weights,
                             data = campaign_wide)
```

Finally, we use the residual balancing weights to fit the MSM from Equation 15:

```
rbwPanel_msm <- svyglm(demprcnt ~ ave_neg * deminc + camp.length +
                         factor(year) + office,
                       design = rbwPanel_design)
```

We report the model results in Table 3. The estimate for nonincumbents is 0.49 and for incumbents is $0.49 - 1.48 = -0.99$. The effect of negative advertisement for nonincumbents is positive though not statistically significant — a ten percentage point increase in the proportion of negative advertising throughout the last five weeks of the campaign increases the Democratic vote share by about half a percentage point, on average. However, the interaction term is significant at the level of 0.01, and incumbents see a sizeable negative effect from negative advertising: a ten percentage point increase in negative advertising reduces a candidate's vote share by about one percentage point, on average.

Table 3: MSM Results for the Cumulative Effect of Negative Advertising on Vote Shares

Average proportion	0.490
	(0.315)
Incumbency	14.971^{***}
	(2.823)
Average proportion * Incumbency	-1.484^{***}
	(0.531)

*p<0.1; **p<0.05; ***p<0.01
Robust standard errors in parentheses.

We conclude by comparing RBW's computational performance to CBPS for a dichotomized version of the treatment representing whether more than 10% of the candidate's advertising was negative for each week. We use this dichotomized variable because CBPS has not been extended to work with continuous treatments in longitudinal settings. Additionally, we do not compare RBW to npCBPS because the latter's use is restricted to point treatment situations. Lastly, though we can construct IPW weights almost immediately, Zhou and Wodtke (2020) show that IPW yields considerably larger effect estimates than RBW and CBPS for this particular case (likely due to the method's susceptibility to model misspecification), so we do not report the IPW results.

Let us first construct the RBW weights. The steps are identical to the ones detailed above for the continuous treatment, the only change being the name of our treatment variable. Next, we use the `CBMSM()` function from the `CBPS` package to generate our CBPS weights. To facilitate comparisons of computation time, we also call `Sys.time()` before and after running the functions that produce the weights in each package. The scalar objects `rbwPanel_time` and `CBPSPanel_time` store how long each method takes to construct the corresponding weights.

```
m1 <-
  lm(dem.polls ~ (d.gone.neg.l1 + dem.polls.l1 + undother.l1) * factor(week),
     data = campaign_long)
m2 <-
  lm(undother ~ (d.gone.neg.l1 + dem.polls.l1 + undother.l1) * factor(week),
     data = campaign_long)

xmodels <- list(m1, m2)

rbwPanel_start <- Sys.time()
rbwPanel_fit <- rbwPanel(
  treatment = d.gone.neg,
  xmodels = xmodels,
  id = id,
  time = week,
  data = campaign_long,
```

```
  future = 0
)

#> Entropy minimization converged within tolerance level

rbwPanel_end <- Sys.time()
rbwPanel_time <- rbwPanel_end - rbwPanel_start

CBPSPanel_form <-
  "d.gone.neg ~ d.gone.neg.l1 + dem.polls + undother + camp.length + deminc + office + factor(year)"

CBPSPanel_start <- Sys.time()
CBPSPanel_fit <-
  CBMSM(
    formula = CBPSPanel_form,
    time = campaign_long$week,
    id = campaign_long$demName,
    data = campaign_long,
    type = "MSM",
    iterations = NULL,
    twostep = TRUE,
    msm.variance = "approx",
    time.vary = TRUE
  )
CBPSPanel_end <- Sys.time()
CBPSPanel_time <- CBPSPanel_end - CBPSPanel_start
```

The output below shows the computation times:

```
rbwPanel_time

#> Time difference of 0.107008 secs

CBPSPanel_time

#> Time difference of 1.059287 mins
```

Whereas RBW takes less than one second to construct the weights, CBPS takes much longer. Hence, the longitudinal setting presents the same pattern we saw above for the point treatment situation: RBW has considerable gains in computational performance over alternative methods of constructing weights for MSMs. Since our focus here is computational performance, we omit the effect estimates for the dichotomized treatment. As shown in Zhou and Wodtke (2020), the results are broadly consistent with those based on the continuous treatment, with RBW and CBPS yielding similar point estimates.

6 Conclusion

Compared to other methods of constructing weights for MSMs, RBW has several advantages. In particular, it does not require modeling the conditional distributions of exposures and is thus easy to use with continuous treatments. Previous simulation studies suggest that it is often more efficient and more robust to model misspecification than alternative weighting strategies (Zhou and Wodtke, 2020). RBW is also favorable from a computational perspective. Its procedure for finding weights involves a convex optimization problem, allowing RBW to locate a solution substantially faster than alternative methods whose optimization relies on nonconvex loss functions — such as the recently proposed nonparametric version of CBPS, which uses the empirical likelihood (Fong et al., 2018). Table 4 sums up these comparisons.

After explaining the underlying logic of RBW, we have described its implementation in the R package **rbw**. With examples from several data sets, we have demonstrated the use of **rbw** in three distinct contexts: effect estimation for point treatments, causal mediation analysis, and effect estimation for time-varying treatments with time-varying confounders.

Nonetheless, RBW is not without limitations. In particular, it depends on models for the conditional means of post-treatment confounders. When these models are incorrectly specified, the pseudo-population generated by residual balancing weights will fail to mimic the original unweighted population, and our estimates will be biased. Even when these models are correctly specified, RBW may also yield biased estimates when we have insufficient balancing conditions. Adding more

Table 4: Comparison of Methods and Software Implementation

Method	Models for Conditional Distributions of Exposures	Balancing Constraints	Implemented for Time-varying Treatments?	R Package
IPW	Required	Absent	Yes	**ipw**
CBPS	Required	Present	Yes	**CBPS**
npCBPS	Not Required	Present	No	**CBPS**
RBW	Not Required	Present	Yes	**rbw**

functions such as cross-products and high-order terms to the set $G(L_t) = \{g_1(L_t), ..., g_{J_t}(L_t)\}$, thereby increasing the number of balancing constraints, may help — though at the risk of making exact balance infeasible. Future work may extend RBW to allow for approximate balance with a penalty for the remaining imbalance in the optimization problem (Fong et al., 2018). Finally, we have relied on the "sandwich" variance estimator to compute standard errors. Though Zhou and Wodtke (2020) demonstrate with simulation studies that this estimator is likely conservative for RBW, they do not provide a variance estimator tailored to address the estimation uncertainty of the RBW weights.

Despite these limitations, RBW will be useful to many social scientists interested in using marginal structural models to study causality in dynamic settings. We will continue to upgrade the package by expanding RBW's ranges of applicability — specifically, to censored data, to contexts involving repeated outcome measures including survival data (Hernán et al., 2000, 2002), and to cases where, as discussed above, exact balance is infeasible and approximate balance must be pursued.

7 Acknowledgments

We thank Geoffrey Wodtke and two anonymous reviewers for their valuable comments on a previous version of this paper.

Bibliography

M. Blackwell. A framework for dynamic causal inference in political science. *American Journal of Political Science*, 57(2):504–520, 2013. ISSN 0092-5853. doi: 10.1111/j.1540-5907.2012.00626.x. [p174, 186]

S. R. Cole and M. A. Hernán. Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, 168(6):656–664, 2008. ISSN 0002-9262. doi: 10.1093/aje/kwn164. [p174]

C. Fong, C. Hazlett, and K. Imai. Covariate balancing propensity score for a continuous treatment: Application to the efficacy of political advertisements. *The Annals of Applied Statistics*, 12(1), 2018. ISSN 1932-6157. doi: 10.1214/17-AOAS1101. [p174, 177, 182, 189, 190]

C. Fong, M. Ratkovic, and K. Imai. *CBPS: Covariate Balancing Propensity Score*, 2021. URL https://CRAN.R-project.org/package=CBPS. R package version 0.22. [p175]

R. B. Geskus and W. M. van der Wal. *ipw: Estimate Inverse Probability Weights*, 2015. URL https://CRAN.R-project.org/package=ipw. R package version 1.0-11. [p174]

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. ISSN 1047-1987. doi: 10.1093/pan/mpr025. [p175, 176, 179]

J. Hainmueller. *ebal: Entropy Reweighting to Create Balanced Samples*, 2014. URL https://CRAN.R-project.org/package=ebal. R package version 0.1-6. [p179]

M. A. Hernán, B. A. Brumback, and J. M. Robins. Marginal structural models to estimate the causal effect of Zidovudine on the survival of HIV-positive men. *Epidemiology*, 11(5):561–570, 2000. ISSN 1044-3983. doi: 10.1097/00001648-200009000-00012. [p190]

M. A. Hernán, B. A. Brumback, and J. M. Robins. Estimating the causal effect of Zidovudine on CD4 count with a marginal structural model for repeated measures. *Statistics in Medicine*, 21(12): 1689–1709, 2002. ISSN 0277-6715. doi: 10.1002/sim.1144. [p190]

K. Imai and I. S. Kim. When should we use unit fixed effects regression models for causal inference with longitudinal data? *American Journal of Political Science*, 63(2):467–490, 2019. ISSN 0092-5853. doi: 10.1111/ajps.12417. [p186]

K. Imai and M. Ratkovic. Covariate balancing propensity score. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 76(1):243–263, 2014. ISSN 1369-7412. doi: 10.1111/rssb.12027. [p174]

K. Imai and M. Ratkovic. Robust estimation of inverse probability weights for marginal structural models. *Journal of the American Statistical Association*, 110(511):1013–1023, 2015. ISSN 0162-1459. doi: 10.1080/01621459.2014.956872. [p174, 186]

K. Imai, L. Keele, and T. Yamamoto. Identification, inference and sensitivity analysis for causal mediation effects. *Statistical Science*, 25(1):51–71, 2010. ISSN 0883-4237. doi: 10.1214/10-STS321. [p184]

K. Imai, L. Keele, D. Tingley, and T. Yamamoto. Unpacking the black box of causality: Learning about causal mechanisms from experimental and observational studies. *The American Political Science Review*, 105(4):765–789, 2011. ISSN 0003-0554. doi: 10.1017/S0003055411000414. [p184]

E. H. Kennedy. *npal: Nonparametric Causal Inference Methods*, 2021. URL https://github.com/ehkennedy/npcausal/blob/master/npcausal.pdf. R package version 0.1.0. [p175]

E. H. Kennedy, Z. Ma, M. D. McHugh, and D. S. Small. Non-parametric methods for doubly robust estimation of continuous treatment effects. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 79(4):1229–1245, 2017. ISSN 1369-7412. doi: 10.1111/rssb.12212. [p175]

R. R. Lau, L. Sigelman, and I. B. Rovner. The effects of negative political campaigns: A meta-analytic reassessment. *The Journal of Politics*, 69(4):1176–1209, 2007. ISSN 0022-3816. doi: 10.1111/j.1468-2508.2007.00618.x. [p174, 186]

A. I. Naimi, E. E. Moodie, N. Auger, and J. S. Kaufman. Constructing inverse probability weights for continuous exposures: A comparison of methods. *Epidemiology*, 25(2):292–299, 2014. ISSN 1044-3983. doi: 10.1097/EDE.0000000000000053. [p174]

J. M. Robins. A new approach to causal inference in mortality studies with a sustained exposure period — Application to control of the healthy worker survivor effect. *Mathematical Modelling*, 7(9): 1393–1512, 1986. ISSN 0270-0255. doi: 10.1016/0270-0255(86)90088-6. [p174, 176]

J. M. Robins. Marginal structural models versus structural nested models as tools for causal inference. In E. M. Halloran and D. Berry, editors, *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, pages 95–133. Springer, New York, 2000. [p174]

J. M. Robins, M. A. Hernán, and B. A. Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11(5):550–560, 2000. ISSN 1044-3983. doi: 10.1097/00001648-200009000-00011. [p174]

M. R. Tomz and J. L. P. Weeks. Public opinion and the democratic peace. *The American Political Science Review*, 107(4):849–865, 2013. ISSN 0003-0554. doi: 10.1017/S0003055413000488. [p174, 184]

C. Urban and S. Niebler. Dollars on the sidewalk: Should U.S. presidential candidates advertise in uncontested states? *American Journal of Political Science*, 58(2):322–336, 2014. ISSN 0092-5853. doi: 10.1111/ajps.12073. [p182, 183]

W. M. van der Wal and R. B. Geskus. ipw: An R package for inverse probability weighting. *Journal of Statistical Software*, 43(13):1–23, 2011. ISSN 1548-7660. [p174]

S. Vansteelandt. Estimating direct effects in cohort and case–control studies. *Epidemiology*, 20(6): 851–860, 2009. ISSN 1044-3983. doi: 10.1097/EDE.0b013e3181b6f4c9. [p174]

Y. Wang, M. Petersen, D. Bangsberg, and M. van der Laan. Diagnosing bias in the inverse probability of treatment weighted estimator resulting from violation of experimental treatment assignment. *U.C. Berkeley Division of Biostatistics Working Paper Series*, Working Paper 211, Sept. 2006. [p174]

X. Zhou and G. T. Wodtke. Residual balancing: A method of constructing weights for marginal structural models. *Political Analysis*, 28(4):487–506, 2020. ISSN 1047-1987. doi: 10.1017/pan.2020.2. [p174, 175, 176, 179, 183, 184, 186, 187, 188, 189, 190]

Derick S. Baum
Harvard University
Department of Sociology
derick_baum@g.harvard.edu

Xiang Zhou
Harvard University
Department of Sociology
xiang_zhou@fas.harvard.edu

Tidy Data Neatly Resolves Mass-Spectrometry's Ragged Arrays

by William Kumler and Anitra E. Ingalls

Abstract Mass spectrometry (MS) is a powerful tool for measuring biomolecules, but the data produced is often difficult to handle computationally because it is stored as a ragged array. In R, this format is typically encoded in complex S4 objects built around environments, requiring an extensive background in R to perform even simple tasks. However, the adoption of tidy data (Wickham, 2014) provides an alternate data structure that is highly intuitive and works neatly with base R functions and common packages, as well as other programming languages. Here, we discuss the current state of R-based MS data processing, the convenience and challenges of integrating tidy data techniques into MS data processing, and present **RaMS**, a package that produces tidy representations of MS data.

1 Introduction

Mass-spectrometry (MS) is a powerful tool for identifying and quantifying molecules in laboratory and environmental samples. It has grown enormously over recent decades and has been responsible for countless advances in chemical and biological fields. It is often paired with liquid chromatography (LC) to separate compounds by retention time and improve detection limits. The large quantity of data produced by increasingly rapid and sensitive instruments has facilitated the adoption of computational methods that use algorithms to detect, identify, and quantify molecular signatures.

Many mass-spectrometrists have some exposure to programming, often in R, and this familiarity is expected to increase in the future as computational methods continue to become more popular and available. However, these researchers typically focus on results and the conclusions that can be drawn from them rather than the arcane details of any particular language or package. This produces a demand for simple data formats that can be quickly and easily understood by even a novice programmer. One such representation is the "tidy" data format, which is rapidly growing in popularity among R users for its consistent syntax and large library of supporting packages (Wickham, 2014). By formatting MS data tidily, the barrier to entry for novice programmers is dramatically reduced, as **tidyverse** functions learned elsewhere will function identically on MS data.

This article begins by reviewing the current theory and implementation of MS data handling, as driven by three major questions. First, why is it difficult to access and interpret MS data? Second, why should it be easier to do this? Finally, why don't current algorithms make it trivial to do this? In the latter portion of this article, we introduce a new package, called R-based access to Mass Spectrometry data (**RaMS**) that provides tidy access to MS data and will facilitate future analysis and visualization.

2 Why is it difficult to access mass-spectrometry data?

Mass spectrometers produce data in the form of ragged (also sometimes called "jagged") arrays. These data structures contain an unequal number of columns per row because any number of ion masses (m/z ratios) may be observed at a given time point. This data is typically managed in a list-of-lists format, with a list of time points each containing a list of the ions observed and their abundances. While this is an effective way to preserve the data structure as it was produced by the instrument, it is less helpful when performing analysis. Typically, analysis (both manual and computational) iterates over m/z windows rather than time. The main focus is the extracted ion chromatogram (EIC) which represents all time points for a given mass, and the spectrum of masses obtained at a given time point is less useful during the preliminary review and initial discovery phases. This nested syntax, often itself contained within S4 objects and encoded as an environment, makes it difficult to extract EICs quickly and intuitively.

Even so, "difficult" is a relative assessment. Veteran R programmers have little difficulty writing elegant code that embraces these ragged arrays and the list-of-lists syntax. Indeed, the dominant MS processing package in R, **MSnbase** currently uses the S4 object system to great effect. However, MS experts are rarely also R experts and have a working familiarity with R rather than a comprehensive background in computer science. This working knowledge typically includes creating plots, subsetting data, and manipulating simple objects but does not extend to the nuances of the S4 object system or methods for rewriting package code. Thus, a package capable of converting these complex data structures into a familiar format appears to be very much in demand.

Finally, it should be noted that existing MS data processing packages are designed to be holistic

pipelines which accept raw data and output definitive results. There is very little room for a user's customization beyond the provided function arguments despite the enormous variability in MS setups, usage, and data quality. It is often challenging to access intermediate objects as a way to debug unexpected results, and published code is rarely easy to edit safely due to poor documentation and unit test coverage. These issues are compounded by the agglomerative nature of R packages that build extensively upon other R packages; the popular **xcms** processing package has over a hundred dependencies installed from across CRAN and Bioconductor, with further functionality provided by unregulated code from GitHub and SourceForge. When combined with additional issues from C++ compilers, versioning, and operating system discrepancies, MS data analysis becomes very much a "black box" with functioning pipelines treated as fragile rather than simple, robust, and reproducible.

3 Why should it be easier to access mass-spectrometry data?

Mass-spectrometry data is fundamentally simple. In LC-MS full-scan mode, each data point has three coordinates corresponding to the time, molecular mass, and intensity dimensions. Even the more complex fragmentation data requires only a single additional dimension, fragment mass. While this ignores the large quantity of critical metadata associated with each file that must also be stored somewhere, a core part of MS research is driven by the data alone. In this preliminary stage of analysis, metadata is less relevant than simple exploratory questions about which molecules can be detected and preliminary assessments of data quality. This exploratory phase is driven by rapid, ad hoc discovery and hypothesis testing that typically requires visualizing chromatograms and the raw data to assess quality: this appears to be one of the reasons why R and its built-in plotting ability is so popular for MS analysis (Gatto et al., 2021). These queries should be trivial to implement, even for beginning R users, but current data storage methods make them difficult and often time-consuming. Currently, the easiest questions to answer about MS data are metadata-based queries about the instrument that the analyst is usually already able to answer. This is an artifact of information storage in most raw data files, with metadata available readily at the top level and measurements buried deep within.

Raw MS data is typically converted from vendor-specific formats into open-source versions that can be parsed without proprietary software. The modern standard is the mzML document, which has been designed to combine the best aspects of precursor standards in a single universal format (Deutsch, 2010). These XML documents have well-defined schema built around a controlled vocabulary to enable consistent parsing. Most critically, the development of the modern mzML format established accession numbers for each attribute which (according to the specification document) should never change. This stability means that the data can be accessed robustly with any XML parser. Older formats, such as mzXML, are currently deprecated and will not undergo further development, making them equally stable.

Finally, simple data formats make it easier to work within existing frameworks rather than developing exclusive functions. Tidy data interacts neatly with the entire **tidyverse** thanks to its shared design philosophy and it's simple to upgrade basic data frames to **data.table**s for improved access speed. More crucially, however, simple formats make it possible to port MS data to other languages and interfaces. It is straightforward to convert an R data frame to Python's pandas version via the **reticulate** package, encode it as a SQL database, or export it as a CSV file to be viewed in Excel or other familiar GUIs. The same cannot be said for R's environments and S4 objects. This connectivity ensures that the best tools possible can be applied to a problem, rather than the subset available in a given package or programming language. Simplifying access to and working storage of MS data is a critical step for the further development of fast, accurate algorithms for the detection and quantification of molecules across many areas of science.

4 Why isn't it already easier to access mass-spectrometry data?

Of course, there are challenges that make simplification difficult and a trade-off must be made between speed, storage, and sanity. Tidy data favors code readability and intuitiveness over computational efficiency: for example, a list-of-lists model is more memory efficient than the proposed rectangular data structure because each time point is stored once rather than repeated in each row. When multiple files are analyzed simultaneously, tidy data also requires that the filename be repeated similarly, resulting in essentially a doubling of object size in the computer memory. Given that most MS experiments involve tens or hundreds of large files, this is a major concern and current packages handle memory carefully, either reading from disk only what is needed or running files in batches. There are several ways to resolve this problem within the tidy data model as well. During the exploration phase, it is rarely necessary to load all data from files simultaneously, but viewing some portion of the data is still critically important for quality control. With the tidy model, it's not

required to import all the data in a single comprehensive step. Instead, quality control files or pooled samples can be viewed as representative of the whole run and rarely challenge memory requirements. Additionally, tidy data makes it easy to subset only the masses of interest for targeted analyses, and the remainder of the data can be discarded from memory. For the final comprehensive analysis, it is much simpler to encode MS data into an external database for access via SQL or other query language when formatted tidily than it is to wrangle current implementations into some accessible object that can handle project sizes larger than the computer's memory.

Theoretically, the ideal data structure for MS data processing speed would invert the current list-of-lists schema by constructing a list of unique *m/z* values, each containing the time points at which that mass ratio was observed and the corresponding intensity. However, this method is complicated by the instrumental error inherent in measuring molecular masses. The same molecule may be measured to have a slightly different mass at each time point, and "binning" these masses together across all time points for a single consensus value risks incorporating nearby masses together even at hypothetical sub-ppm mass accuracy (Kind and Fiehn, 2006). Instead, *m/z* values are continuous rather than discrete, making it difficult to encode the data in this way. A tidy framework resolves part of this issue by storing the time and *m/z* values in columns that can be indexed by a binary search, such as the one implemented by **data.table**. This allows for rapid subsetting by both time and *m/z*. Finally, it is worth noting that computers have rapidly grown faster and larger while human intuition has not grown as quickly. This indicates that concerns with processing time and memory will lessen over time and that in the long run, sanity should be prioritized over speed and storage.

There are other reasons that a tidy approach has not yet been implemented for MS data. MS files include large amounts of metadata which should not be discarded, but are challenging to encode efficiently in a rectangular format. A proper tidy approach requires that a separate table be constructed to hold this per-file metadata, with a key such as file name that permits joining the metadata back to the original information. Compared to the monolithic S4 objects constructed by traditional workflows, managing multiple tables may be unappealing. S4 objects also excel at recording each process that is performed on the data, and a specific "processes" slot is found in some objects to record exactly this. However, with the emergence of code sharing and open-source projects it becomes less critical that the data itself records the process because the source code is available.

Finally, a significant history exists for today's methods. **MSnbase**, the first widely-used R package designed to process MS data, implemented S4 objects as a way to hold entire MS experiments in memory, and dependent packages extend this MSnExp object in various ways rather than discarding it entirely. This development history and connected network of packages is incredibly useful and represents an extensive process of innovation and refinement. We would like to emphasize that the concerns raised here and the package introduced below are not designed to critique or replace this significant effort. Instead, our goal is to function alongside prior work as a way to enable rapid, interactive, and preliminary exploration. Following initial investigation, we recommend using the existing pipelines and extensive package network to establish a reproducible, scripted process of MS data analysis.

5 The RaMS package

The **RaMS** package implements in R a set of methods used to parse open-source mass-spectrometry documents into the R-friendly data frame format. Functions in the package accept file names and the type of data requested as arguments and return rectangular data objects stored in R's memory. This data can then be processed and visualized immediately using base R functions such as plot and subset, passed to additional packages such as **ggplot2** and **data.table**, or exported to language-agnostic formats such as CSV files or SQL databases.

Installation

The **RaMS** package can be installed in two ways:

The release version from CRAN:

```
install.packages("RaMS")
```

Or the development version from GitHub:

```
# install.packages("remotes")
remotes::install_github("wkumler/RaMS")
```

Input arguments

RaMS is simple and intuitive, requiring the memorization of a single new function `grabMSdata` with the following usage:

```
grabMSdata(files)
```

Where `files` is a vector of file paths to mzML or mzXML documents, which can be located on the user's computer, a network drive, FTP site, or even at a URL on the Internet. Further parameters are documented below in Table 1:

Parameter	Description
`grab_what`	Specifies the information to extract from the mzML or mzXML file. Can currently accept any combination of "MS1", "MS2", "EIC", "EIC_MS2", "metadata", and "everything" (the default).
`verbosity`	Controls progress messages sent to the console at three different levels: no output, loading bar and total time elapsed, and detailed timing information for each file.
`mz`	Used when `grab_what` includes "EIC" or "EIC_MS2". This argument should be a vector of the *m/z* ratios interesting to the user, if the whole file is too large to load into memory at once or only a few masses are of interest.
`ppm`	Used alongside the `mz` argument to provide a parts-per-million error window associated with the instrument on which the data was collected.
`rtrange`	A length-two numeric vector with start and end times of interest. Often only a subset of the LC run is of interest, and providing this argument limits the data extracted to those between the provided bounds.

Table 1: Parameters accepted by the `grabMSdata` function.

Usage

Extracting data with `grabMSdata` returns a list of tables, each named after one of the parameters requested. A `grab_what` argument of `"MS1"` will return a list with a single entry, the MS^1 (i.e. full-scan data) for all of the files:

```
msfile <- system.file("extdata", "LB12HL_AB.mzML.gz", package = "RaMS")
msdata <- grabMSdata(files = msfile, grab_what="MS1")
head(msdata$MS1)
```

rt	mz	int	filename
4.009	104.0710	1297755.000	LB12HL_AB.mzML.gz
4.009	104.1075	140668.125	LB12HL_AB.mzML.gz
4.009	112.0509	67452.859	LB12HL_AB.mzML.gz
4.009	116.0708	114022.531	LB12HL_AB.mzML.gz
4.009	118.0865	11141859.000	LB12HL_AB.mzML.gz
4.009	119.0837	9636.127	LB12HL_AB.mzML.gz

Table 2: Tidy format of RaMS output showing columns of MS^1 data, with columns for retention time (rt), mass-to-charge ratio (mz), intensity (int) and name of the source file (filename). Note that this is a subset - the actual object contains 8,500 entries.

This table is already tidied, ready to be processed and visualized with common base R or **tidyverse** operations. For example, it's often useful to view the maximum intensity observed at each time point: this is known as a base peak chromatogram or BPC. Below are two examples of calculating and plotting a BPC using base R and the **tidyverse**.

```
# Base R
BPC <- tapply(msdata$MS1$int, msdata$MS1$rt, max)
plot(names(BPC), BPC, type="l")
```

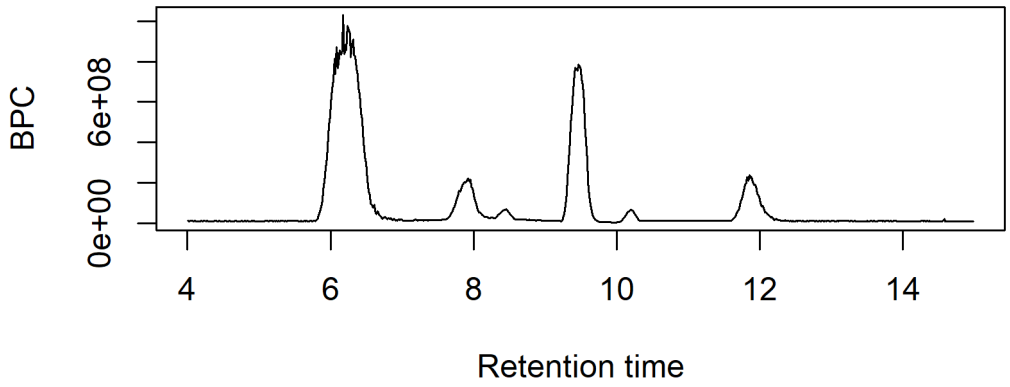


Figure 1: A simple chromatogram plotted using base R. This plot shows the retention time of all compounds in a sample plotted against the maximum intensity at each timepoint. Base graphics were used so the plot is fully customizable with normal graphics options.

```
# Tidyverse
library(tidyverse)
BPC <- msdata$MS1 %>%
    group_by(rt) %>%
    summarize(BPC_int=max(int))
ggplot(BPC) + geom_line(aes(x=rt, y=BPC_int))
```

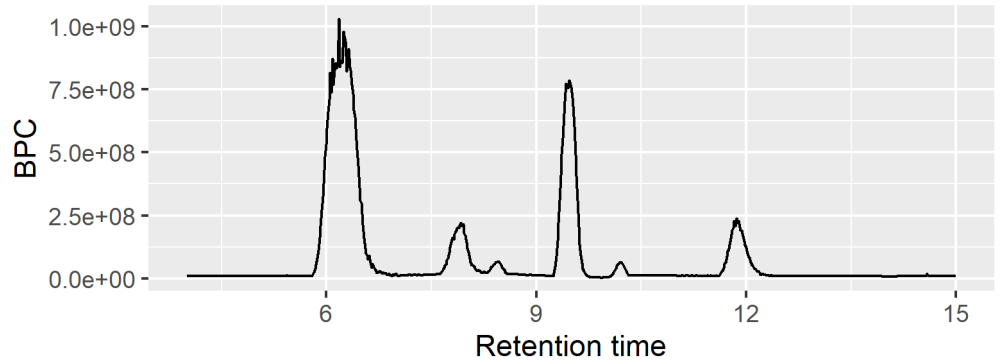


Figure 2: A simple chromatogram plotted using the **ggplot2** package. This plot shows the same data as Figure 1 of retention time by maximum intensity across compounds but uses **ggplot2** syntax and defaults.

Importantly, note that the creation of these plots required no special knowledge of the S3 or S4 systems and the plots themselves are completely customizable. While similar packages provide methods for plotting output, it is rarely obvious what exactly is being plotted and how to customize those plots because the data is stored in environments and accessed with custom code. **RaMS** was written with the beginning R user in mind, and its design philosophy attempts to preserve the most intuitive code possible.

RaMS uses **data.table** internally to enhance speed, but this also allows for more intuitive subsetting in mass-spectrometry data. With **data.table**, operations are nearly as easy to write in R as they are to write in natural language, leveraging the user's intuition and decreasing the barrier to entry for non-coder MS experts. For example, a typical request for MS data might be written in natural language as:

> "All MS^1 data points with *m/z* values between an upper and lower bound, from start time to end time."

This request can be written in R almost verbatim thanks to **data.table**'s intuitive indexing and `%between%` function:

```
msdata$MS1[mz %between% c(upper_bound, lower_bound) &
    rt %between% c(start_time, end_time)]
```

Most importantly, this syntax doesn't require the mass-spectrometrist to have an understanding of how the data is stored internally. Current implementations use S4 objects with slots such as "chromatograms" and "spectra" or derivatives of these, despite their inconsistent usage across the field and unclear internal structure. (Smith et al., 2015)

RaMS enhances the intuitive nature of **data.table**'s requests slightly by providing the pmppm function, short for "plus or minus parts-per-million (ppm)". Masses measured on a mass-spectrometer have a certain degree of inherent deviation from the true mass of a molecule, and the size of this error is a fundamental property of the instrument used. This means that mass-spectrometrists are often interested in not only the data points at an exact mass, but also those within the ppm error range. MS data exploration often makes requests for data in natural language like:

> "All MS^1 data points with *m/z* values within the instrument's ppm error of a certain molecule's mass"

Which can again be expressed in R quite simply as:

```
msdata$MS1[mz %between% pmppm(molecule_mass, ppm_error)]
```

Internals

Fundamentally, **RaMS** can be considered an XML parser optimized for mzML and mzXML documents. The rigorous specification and detailed documentation make it possible for a generic XML parser to efficiently extract the document data. In R, the **xml2** package provides modern parsing capabilities and is efficient in both speed and memory usage by calling C's libxml2 library, making it an attractive choice for this processing step. Much of **RaMS**'s internal code consists of a library of XPath expressions used to access specific nodes and extract the (often compressed) values . Table 3 below provides several examples of XPath expressions used to extract various parameters from the mzML internals:

Parameter of interest	mzML XPath expression
Fragmentation level	//spectrum/cvParam[@name="ms level"]
Retention time	//scanList/scan/cvParam[@name="scan start time"]
m/z values	//binaryDataArrayList/binaryDataArray[1]/binary
Intensity values	//binaryDataArrayList/binaryDataArray[2]/binary
Polarity (for positive mode)	//spectrum/cvParam[@accession="MS:1000130"]

Table 3: A few example parameters extracted from the mzML file and the corresponding XPath expression used to extract it.

These sample expressions illustrate the controlled vocabulary of the mzML parameters (the cvParam elements above) and the remarkable stability of the specification that permits optimization. While the "polarity" parameter for positive mode is the only one above that is specified via its accession number ("MS:1000130"), it's worth noting that the other parameters also have unique accession number attributes that could be used but instead have been foregone in favor of readability.

MS data files are often highly compressed and the *m/z* and intensity data is typically encoded as base 64 floating point arrays. MS data extracted from the binary data array must then first be decoded from base64 to binary using the **base64enc** package, then decompressed if necessary using R's base `memDecompress` function, and finally cast to double-precision floating point values via base R's `readBin`.

After the data has been extracted from the XML document, **RaMS** uses the **data.table** package to provide fast aggregation and returns **data.table** objects to the user. This is also the step which converts the data from a ragged array format into a tidy format, and neatly illustrates the strength of tidy data. Rather than continuing to store the data as a list-of-lists and preserving the nested data structure, this step creates separate columns for retention time (rt) and *m/z* (mz) values. This allows the user to perform rapid binary searches on both the retention time and *m/z* columns and can greatly accelerate the extraction of individual masses of interest, as is often the goal when analyzing MS data.

Comparison to similar packages

While many packages exist to process MS data within R, very few can be found that actually read the raw data into the R environment. The dominant package by far is **MSnbase**, which describes itself as

providing "infrastructure for manipulation, processing and visualisation of mass spectrometry and proteomics data", and is thus very similar to **RaMS**. **MSnbase** itself calls the Bioconductor package **mzR** to provide the C++ backend used to parse the raw XML data. Other packages include **readMzXmlData** and **MALDIquantForeign**, both developed by Sebastian Gibb and hosted on CRAN. One additional package to note is the **caMassClass** package that no longer exists on CRAN but code from which can be found in the **CorrectOverloadedPeaks** package and only parses the deprecated mzXML format. Finally, the **Spectra** package is under active development by the RforMassSpectrometry initiative and represents a useful comparison for other cutting-edge frameworks that will be expanded in the future (Rainer et al., 2022). However, all of these packages preserve the list-of-list format and none produce naturally tidy representations.

This section illustrates how **RaMS** compares to **MSnbase** as the current dominant processing package and **Spectra** as the next iteration of MS processing. **MSnbase** has undergone constant revision since its inception in 2010, while **Spectra** has been under development since 2020. The most recent version of **MSnbase** as of this writing was announced in 2020 and focuses on the new "on-disk" infrastructure that loads data into memory only when needed. This new infrastructure and the legacy storage mode released in the first version of **MSnbase** provide useful comparisons for **RaMS** in terms of memory usage and speed and the **Spectra** package will provide a useful future-oriented comparison. As noted above, however, **RaMS** has different goals from either of these packages. **RaMS** is optimized for raw data visualization and rapid data exploration while **MSnbase** and **Spectra** are designed to provide a solid foundation for more streamlined data processing and these packages all can work neatly in concert rather than replacing each other.

To compare the different methods, ten MS files were chosen from the MassIVE dataset MSV000080030 to mimic the large-experiment processing of Gatto et al. (2021). Methods were compared in terms of memory usage, time required to load the data into R's working memory, and the time required to subset an EIC and plot the data. Due to the differences in method optimization, we expected **MSnbase** to be significantly faster when loading the data, **RaMS** to be significantly faster during subsetting and plotting, and **MSnbase** to have the smallest memory footprint. The **Spectra** package's capabilities were less well known in advance but should represent a consistent improvement over **MSnbase**. These expectations were well-validated by the results shown in Figure 3.

RaMS performed better than expected on the data load-time metric, taking approximately the same amount of time as the new on-disk **MSnbase** backend and the **Spectra** package and significantly less than the old in-memory method. This was surprising because while **RaMS** is performing the physical I/O process essentially equivalent to the creation of the MSnExp, both the OnDiskMSnExp method and the Spectra object instead create a system of pointers to the data and don't actually read the data into memory. However, the new backend begins to perform better as the number of files increases and proportional improvements are expected with even larger file quantities. The **Spectra** package, as expected, shows consistent improvements over both **MSnbase** backends.

For the subsetting and plotting metric, our expectation that **RaMS** would be the fastest method was validated by times approximately two orders of magnitude smaller than those obtained by **MSnbase** (note the log scale used in the figure). These results also validated earlier results demonstrating the superiority of the new on-disk method (Gatto et al., 2021) and the improvements in the new **Spectra** package. The sub-second subset and plot times of **RaMS** are so much smaller than the other timings recorded in this trial that **RaMS** essentially has a single fixed cost associated with the initial data import, making it ideal for the exploratory phase of data analysis where files are loaded once and then multiple chromatograms may be extracted and reviewed. This design also aligns with the user's expected workflow in which data import is accepted as a time-consuming task, but subsequent analysis should be relatively seamless and instantaneous.

The greatly reduced subsetting and plotting time required by **RaMS** and the observation that file load times and data plotting times were approximately equal for MSnbase led to the creation of the bottom-left graph in Figure 3. This follow-up analysis highlights that the slightly increased file load time of **RaMS** combined with the very short subsetting and plotting phase is actually less than the total time required by **MSnbase** and **Spectra** to read, subset, and plot, establishing **RaMS** as the fastest option even if the end goal is to extract a single chromatogram. This follow-up also demonstrates the largest improvements of the new **MSnbase** on-disk method over the old one and the clearest improvements in **Spectra**.

As expected, this speed comes at a cost. **RaMS** has a larger memory footprint than even the old in-memory MSnExp object. While all three objects grew approximately linearly with the number of files processed, the **RaMS** object was approximately 2 times larger than the in-memory **MSnbase** object and several orders of magnitude larger than the new, on-disk version. This was expected because **RaMS** stores retention time and filename information redundantly in the tidy format while the list-of-lists method only stores that information once. In fact, the **RaMS** object size was larger than the uncompressed mzXML files themselves! However, this trade-off can be minimized through the

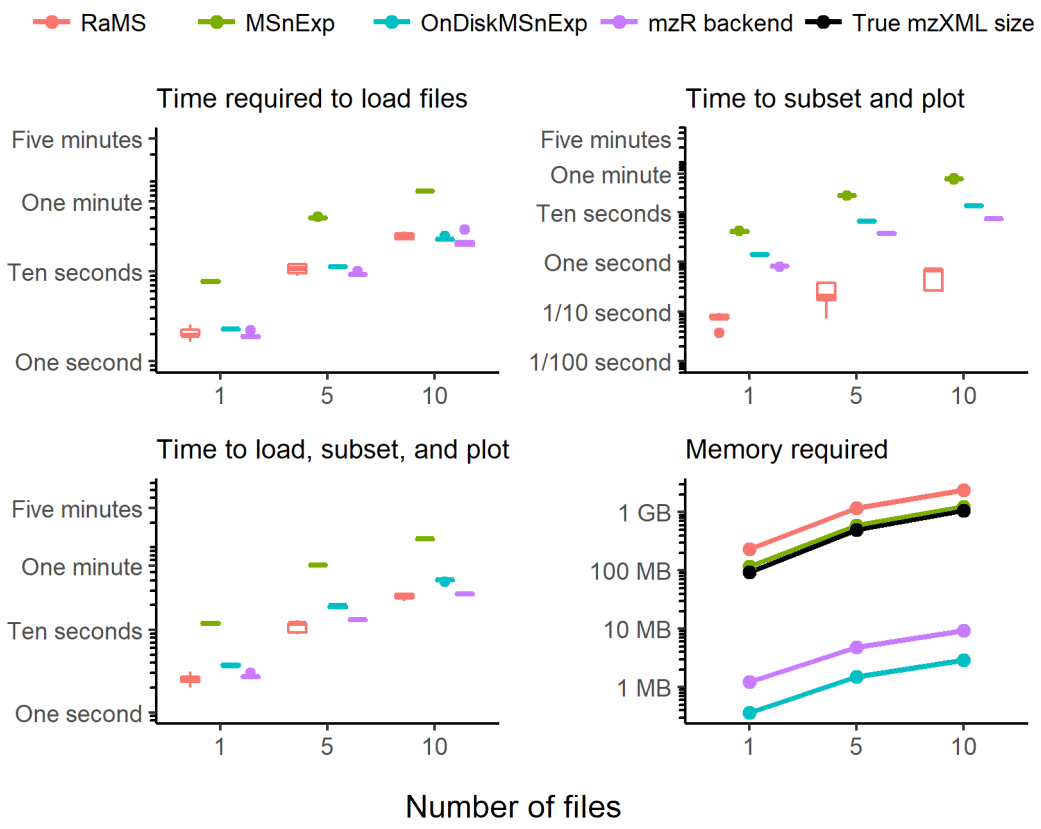


Figure 3: Time and memory required by **RaMS** compared to the **MSnbase** and **Spectra** methods across 1, 5, and 10 mzXML files. The top-left plot shows the time required to load the mzXMLs into memory (**RaMS** and **MSnExp**) or construct pointers (OnDiskMSnExp, **Spectra**'s mzR backend) with the MSnExp object taking approximately an order of magnitude longer than the other methods. The top-right plot shows the time required to subset the data by *m/z* to a single chromatogram and plot that subset after the object has already been created. The **RaMS** package performs this approximately an order of magnitude faster than the other packages and the **Spectra** package is second-fastest, with **RaMS** taking less than a second for up to 10 mzXMLs and the **Spectra** package taking between one and ten seconds depending on the number of files to be subset. The bottom-left plot shows a combination of the two plots above by timing each package as it performs the full object construction, subsets to a single chromatogram, and plots it with **RaMS** again the fastest among the packages. The bottom-right plot shows the memory required for each package across different numbers of files as well as the size of the original mzXML documents as a benchmark. Both **RaMS** and the MSnExp objects occupied more space in RAM than the original file size (**RaMS** occuying approximately 2x as much memory, MSnExp closer to 1.1x), while the OnDiskMSnExp and mzR backend were consistently two orders of magnitude smaller. Times were obtained by the **microbenchmark** package and object sizes were obtained with **pryr**. Note the log-scaled y-axes.

use of **RaMS**'s vectorized `grab_what = "EIC"` and `grab_what = "EIC_MS2"` functions that can extract a vector of masses of interest and discard the remainder of the data to free up memory for analyses where the specific ions of interest are known beforehand. The general lesson from this analysis seems to be that if the memory is available and a quick and intuitive interaction is desired, **RaMS** is now the top contender. For other purposes, **MSnbase** or **Spectra** remain the obvious choices depending on expected workflow.

Broader interactions

RaMS is intentionally simple. By encoding MS data in a rectangular, long data format, **RaMS** facilitates not only R-specific development but contributes to MS analysis across languages and platforms. At the most basic level, subsets of interest can be exported as CSV files for use in any language that can read this ubiquitous format. Even users with zero programming background are familiar with Excel and other spreadsheet GUIs, so this method of export and data-sharing improves transparency by

allowing anyone to open the raw data corresponding to compounds of interest.

The list-of-tables format that **RaMS** returns was inspired by traditional relational databases, and this provides a slightly more complex method of storing data with several advantages over CSV export. The dominant convenience of relational databases is that they can grow almost indefinitely, rather than being limited by computer memory. While existing packages perform admirably when operating on files that fit into RAM, there are few good solutions for the MS experiments that can exceed hundreds of gigabytes in size. Both batching and subset analysis face issues with systematic inter-sample variation rarely controlled for across subsets. Additionally, an external relational database can be easily appended with additional files as experiments continue to be performed, rather than demanding that all samples be run before any analysis can begin. **RaMS** output can be easily written to SQL databases using existing packages such as **DBI** and **RSQLite**:

```
library(DBI)
db <- dbConnect(RSQLite::SQLite(), "msdata.sqlite")
dbWriteTable(db, "MS1", msdata$MS1)
dbListTables(db)
dbGetQuery(db, "SELECT * FROM MS1 LIMIT 3")
dbDisconnect(db)
```

Finally, with **reticulate**, R data frames can be directly coerced into Pandas DataFrames. This allows for an unprecedented degree of interaction between R and Python for MS data analysis, reducing the need for parallel development in both languages and allowing the optimal functions to be used at each step rather than the limited selection that have already been implemented in R or Python. As MS data exploration and analysis continues to grow increasingly machine-learning heavy, allowing R to interact elegantly with Python enables the best of R's extensive MS analysis history with Python's powerful interfaces to deep learning frameworks such as TensorFlow and Pytorch.

6 Summary

In this paper, we discussed the current paradigm of MS data analysis in R and identify an area where tidy data techniques significantly improve user experience and support increased interaction with other packages and software. We also present **RaMS** as a package that fills this gap by presenting MS data to the R user in a tidy format that can be instantly queried and plotted.

7 Acknowledgements

We are grateful to members of the Ingalls Lab and other labs at the University of Washington who gave invaluable feedback on early versions of this package and the philosophy behind it. Katherine Heal and Laura Carlson generated the data used in the demo files and were early adopters, and Angie Boysen and Josh Sacks provided crucial testing and application of the package. We also thank both anonymous reviewers for their insightful commentary and suggestions that improved both the manuscript and the CRAN package. This work was supported by grants from the Simons Foundation (329108, 385428, and 426570, A.E.I.).

Bibliography

E. W. Deutsch. Mass spectrometer output file format mzML. In S. J. Hubbard and A. R. Jones, editors, *Proteome Bioinformatics*, pages 319–331. Humana Press, Totowa, NJ, 2010. ISBN 978-1-60761-444-9. doi: 10.1007/978-1-60761-444-9_22. URL https://doi.org/10.1007/978-1-60761-444-9_22. [p194]

L. Gatto, S. Gibb, and J. Rainer. MSnbase, efficient and elegant R-based processing and visualization of raw mass spectrometry data. *Journal of Proteome Research*, 20(1):1063–1069, 2021. doi: 10.1021/acs.jproteome.0c00313. URL https://doi.org/10.1021/acs.jproteome.0c00313. [p194, 199]

T. Kind and O. Fiehn. Metabolomic database annotations via query of elemental compositions: mass accuracy is insufficient even at less than 1 ppm. *BMC Bioinformatics*, 7(1):234, Apr 2006. doi: 10.1186/1471-2105-7-234. URL https://doi.org/10.1186/1471-2105-7-234. [p195]

J. Rainer, A. Vicini, L. Salzer, J. Stanstrup, J. M. Badia, S. Neumann, M. A. Stravs, V. Verri Hernandes, L. Gatto, S. Gibb, and M. Witting. A modular and expandable ecosystem for metabolomics data

annotation in R. *Metabolites*, 12(2), 2022. ISSN 2218-1989. doi: 10.3390/metabo12020173. URL https://doi.org/10.3390/metabo12020173. [p199]

R. Smith, R. M. Taylor, and J. T. Prince. Current controlled vocabularies are insufficient to uniquely map molecular entities to mass spectrometry signal. *BMC Bioinformatics*, 16(7):S2, Apr 2015. doi: 10.1186/1471-2105-16-S7-S2. URL https://doi.org/10.1186/1471-2105-16-S7-S2. [p198]

H. Wickham. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. doi: 10.18637/jss.v059.i10. URL https://doi.org/10.18637/jss.v059.i10. [p193]

William Kumler
University of Washington School of Oceanography
1501 NE Boat St., Seattle, WA 98105 United States of America
ORCiD: 0000-0002-5022-8009
wkumler@uw.edu

Anitra E. Ingalls
University of Washington School of Oceanography
1501 NE Boat St., Seattle, WA 98105 United States of America
ORCiD: 0000-0003-1953-7329
aingalls@uw.edu

Log Likelihood Ratios for Common Statistical Tests Using the likelihoodR Package

by Peter Cahusac

Abstract The **likelihoodR** package has been developed to allow users to obtain statistics according to the likelihood approach to statistical inference. Commonly used tests are available in the package, such as: *t* tests, ANOVA, correlation, regression and a range of categorical analyses. In addition, there is a sample size calculator for *t* tests, based upon the concepts of strength of evidence, and the probabilities of misleading and weak evidence.

1 Introduction

Maximum likelihood estimation (MLE) is well-understood and widely used throughout statistics. In contrast, the use of the likelihood function as a basis for inference is much less understood and even confused with MLE. As Edwards wrote in his excellent book : "At one recent international conference at which I laboured for three-quarters of an hour to make clear the advantages of likelihood inference, the chairman thanked me for my lecture on the Method of Maximum Likelihood" (Edwards, 1992) p 101. In the Epilogue of this book on p 212, Edwards says that MLE is "a red herring". To clarify: MLE is used to estimate a parameter value according to a supposed probability distribution, while likelihood inference is used to compare two hypotheses through the ratio of their values on a likelihood function.

All the main statistical approaches to scientific inference (frequentist, Bayesian and information criterion) are based upon calculated probabilities. The frequentist approach, for example, typically uses a sampling distribution centred on a null hypothesis and calculates the probability of obtaining the observed value or values more extreme from the null. The likelihood approach, also known as the evidential approach, differs in that it is simply based upon the evidence provided by the observed data (not including more extreme values) and represented by the likelihood function. The ratio of the heights under the likelihood function according to the hypothesis values tested provides the likelihood ratio. To create a linear scale of evidence, the natural logarithm of this is taken to give us the log likelihood ratio, also known as the support, a term that was first defined by Harold Jeffreys (Jeffreys, 1936).

The likelihood approach is not subject to the same criticisms often leveled at the frequentist and Bayesian approaches (Goodman and Royall, 1988; Edwards, 1992; Royall, 1997; Goodman, 1999; Dixon, 2003; Dienes, 2008; Wasserstein and Lazar, 2016; Lakens, 2021). Likelihood ratios provide objective measures of evidence between competing hypotheses, unaffected by the intentions of the investigator. Log likelihood ratio (*support*, see below) values are proportional to the quantity of data, representing the weight of evidence. This means that support values from independent studies can simply be added together, e.g. for meta-analysis. Unlike other approaches based on probabilities, likelihood ratios are unaffected by transformations of variables.

Despite the apparent advantages of the evidential approach there is a dearth of available resources in statistical computing. None of the major commercial packages (e.g. SPSS, SAS, Minitab) provide likelihood ratios or support values. This should not be confused with the wide availability of likelihood ratio tests (also known as G-tests), which ultimately provide *p* values according to the frequentist approach. Because virtually no software is available for analysis, this impacts on the use of the evidential approach in scientific reporting. This also impacts on the teaching of the evidential approach, which then negatively feeds back to reduced scientific reporting. The **likelihoodR** package for R (Cahusac, 2021) is an attempt to address this situation and calculations are based upon the recent book (Cahusac, 2020b). R (Ihaka and Gentleman, 1996) is a widely used statistical platform with a huge variety of packages.

2 Support

The package always reports the support (log likelihood ratio). This relative evidence scale ranges from -∞ to +∞, with zero representing no evidence either way. A working interpretation for this scale has been offered by (Goodman and Royall, 1988), see Table 1. No threshold need be applied, and S values are given to just one decimal place, e.g. 2.3.

S	**Interpretation of H_1 vs H_2**
0	No evidence either way
1	Weak evidence
2	Moderate evidence
3	Strong evidence
4	Extremely strong evidence

Table 1: Interpretation for values of S, the support, calculated as the natural logarithm of the likelihood ratio. Negative values would represent support for hypothesis values H_2 vs H_1. Typically, it is sufficient to give S to one decimal place.

The support (S) values reported in the package are distinct from the surprise/surprisal S-values described by (Palm, 2012) based on Shannon information theory, and that by (Greenland, 2019) produced by simply taking the negative log base 2 of the p value.

The likelihood intervals are reported wherever possible and these are given in terms of support rather than likelihood ratio. A typical likelihood interval (support interval) is the S-2 interval, which numerically often closely corresponds to the frequentist 95% confidence interval (Cahusac, 2020b). The S-2 interval represents an interval based upon $e^{-2} = 0.135 = 1 / 7.40$ likelihood ratio interval. All the values within the interval would have a likelihood ratio of no less than 1 / 7.40, or an S = -2. The S-3 interval would be $e^{-3} = 0.05 = 1 / 20.09$, and so on. Analyses also report other relevant statistics, such as t, F and χ^2, as well as the corresponding frequentist p value. Where possible the likelihood function is given, decorated with hypothesis parameter values shown as coloured lines. Currently there are 14 different statistical tests implemented by the package, the summary for each are given in Table 2 (listed alphabetically).

Function	**Description**
`L_1way_ANOVA`	Independent samples one-way ANOVA
`L_1way_cat`	One-way categorical data analysis for binomial and multinomial
`L_1way_RM_ANOVA`	One-way repeated measures ANOVA
`L_2S_ttest`	Independent samples t test
`L_2way_cat`	Two-way categorical data analysis
`L_2way_Factorial_ANOVA`	Two-way independent samples factorial ANOVA
`L_corr`	Bivariate normal correlation
`L_efficacy`	Efficacy analysis for binomial categorical data
`L_logistic_regress`	Multiple logistic regression
`L_OR`	Odds ratio
`L_regress`	Bivariate regression for linear, quadratic and cubic comparisons
`L_RR`	Relative risk
`L_ttest`	One sample and related samples t test
`L_t_test_sample_size`	Sample size calculation using the evidential approach for t tests

Table 2: A summary of the functions available in the **likelihoodR** package

3 Continuous data

A range of tests for differences of continuous variables is available. One function `L_ttest` performs a one sample and related samples test. As well as specifying the null value, two alternative hypothesis values can be specified, one in terms of the measurements used and the other in terms of Cohen's d.

```
L_ttest(data1, data2, null=0, d=0.5, alt.2=NULL, L.int=2, verb=TRUE)
```

Where the arguments are:

data1	a (non-empty) numeric vector of data values
data2	a (non-empty) numeric vector of data values for related sample, default = NULL
null	value for the null hypothesis, default = 0
d	Cohen's effect size, default = 0.5
alt.2	value for an alternative hypothesis, in units used for data, default = NULL
L.int	likelihood interval given for a given support value, e.g. 2 or 3, default = 2
verb	show output, default = TRUE

As an example:

```
> # one sample Gosset's original additional hours of sleep data Cahusac (2020) p 29
> mysample <- c(0.7, -1.6, -0.2, -1.2, -0.1, 3.4, 3.7, 0.8, 0.0, 2.0)
> a1=L_ttest(mysample, d=0.5, alt.2=2, L.int=2)

  Maximum support for the observed mean 0.75 (dashed line) against the null 0 (black line) = 0.892
Support for d of 0.5 (0.8945048, blue line) versus null = 0.856
Support for d versus 2nd alt Hypothesis 2 (green line) = 2.131
Support for 2nd alt Hypothesis versus null = -1.275

S-2 likelihood interval (red line) is from -0.44025 to 1.94025

t(9) = 1.326, p = 0.2175978, d = 0.419
```

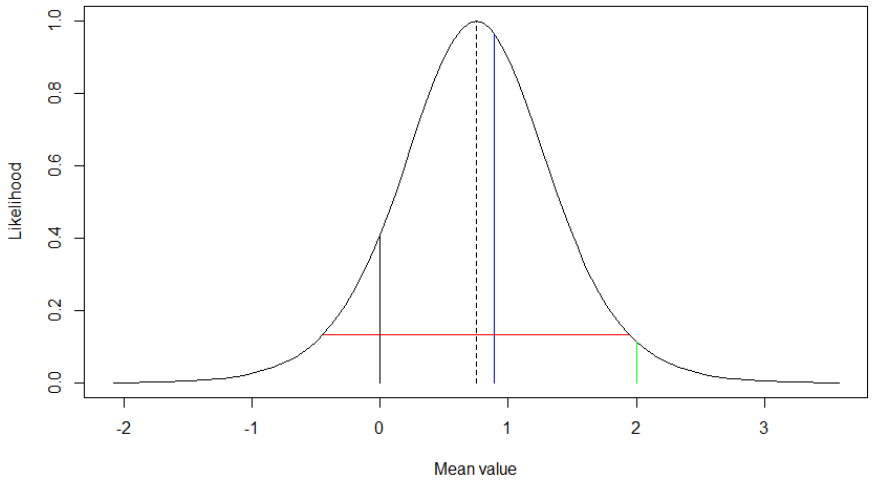


Figure 1: Graphical output from running the one-sample test function `L_ttest`. The likelihood function for the mean value given the data, with the MLE (sample mean) represented by the vertical dashed line. The black line is the specified null value (here at 0), the blue line is for a specified alternative hypothesis (for an effect size of 0.5), and the green line represents a second alternative hypothesis (for 2 hours). The red horizontal line shows the *S*-2 likelihood interval.

The assigned object `a1` contains values of interest which can be assessed in the usual way with a1$..., but entering `a1` on its own gives all the values:

`$obs.mean`	the observed mean or difference in mean for related samples
`$df`	degrees of freedom
`$alt.H1`	mean value according to specified d
`$alt.H2`	specified second hypothesis value
`$S_max`	maximum support for observed mean against the null
`$S_10`	support for d versus null
`$S_12`	support for d versus specified second hypothesis
`$S_20`	support for second hypothesis versus the null
`$like.int`	likelihood interval
`$L.int.spec`	specified likelihood interval in units of support
`$null.value`	null value
`$t.val`	t value for test against null
`$p.val`	*p* value for test against null
`$d.obs`	observed effect size

For the related sample test a second vector of values of equal length to data1 (and paired) is entered for `data2`. The independent samples test function is `L_2S_ttest`. The data is entered as the first argument followed by the vector of the same length coding groups. Similar arguments to the one sample/related sample test can be specified and similar output, but this time showing the difference in means.

The package includes one-way ANOVA (`L_1way_ANOVA`), one-way repeated measures ANOVA (`L_1way_RM_ANOVA`) and two-way between-participants factorial ANOVA (`L_2way_Factorial_ANOVA`). All these functions use contrasts, employing the model comparison approach espoused by Glover and Dixon (Dixon, 2003; Glover and Dixon, 2004; Dixon, 2013). For the one-way analyses if the arguments for the contrasts are not specified then they default to testing a linear and a quadratic contrast. In the factorial ANOVA, no contrast comparisons are made if the contrasts are left unspecified. If the first contrast is specified then this is compared to the main effects, and if the second contrast is specified then this is compared to the first contrast. Such contrast comparisons are easy to do in this approach, and are difficult or impossible to do using frequentist *p* values. The support values for the between-participants analyses are adjusted using Akaike's correction (Hurvich and Tsai, 1989). We will look at `L_2way_Factorial_ANOVA`. The first line of output gives the S for comparing the full model of main effects and interaction with the null model. Using data from Cahusac (2020b) p 91

```
> time <- c(6.4, 4.6, 6.4, 5.6, 5.9, 6.1, 6.3, 4.5, 4.8, 6.6, 7, 9.3, 7.9, 9.4, 8.2,
          4.4, 4.2, 5, 6.9,4.5, 4, 4.3, 6.9, 5.5, 5.8, 4.4, 4.2, 5.1, 6.9, 4.5)
> Treatment = gl(3,5,30, labels=c("T1","T2","T3"))
> Health = gl(2,15,30, labels=c("Hemophiliac","Normal"))
> contrast1 <- c(-1, -1, 5, -1, -1, -1) # interaction Hemo T3 higher than others
> contrast2 <- c(-1, -1, -1, 1, 1, 1)    # main effect of health status (Hemo higher)
> m1=L_2way_Factorial_ANOVA(time, Treatment, Health, contrast1, contrast2, verb=TRUE)

Support for full model (including interaction) versus null = 7.036
 Support for full model versus main effects = 2.968
 Support for contrast 1 versus main effects = 7.587
 Support for contrast 1 versus contrast 2 = 8.462

First factor main effect F(2,24) = 5.039, p = 0.01488452, partial eta-squared = 0.296
 Second factor main effect F(1,24) = 15.992, p = 0.0005281828, partial eta-squared = 0.4
 Interaction F(2,24) = 6.219, p = 0.006664992, partial eta-squared = 0.341
 Contrast 1 F(1,24) = 36.048, p = 3.373928e-06
```

As before, the assigned object `m1` contains values of interest which can be assessed in the usual way with `m1$...`, but entering `m1` on its own gives all the values.

Bivariate normal correlation uses the `L_corr function`, and is demonstrated by using data from the heptathlon (Cahusac, 2020b) p 104

```
> m200 <- c(22.6,23.7,23.1,23.6,23.6,23.6,25.5,23.9,24.5,23.9,24.9,24.8,24.7,
          25.0,24.6,24.9,25.0,25.6,24.8,25.5,25.7,24.9,26.6,25.2,26.2)
> m800 <- c(128.5,126.1,124.2,132.5,134.7,132.5,138.5,127.9,133.7,132.2,136.1,142.8,
          125.8,131.5,137.1,134.9,146.7,133.9,146.4,144.0,133.4,138.0,139.2,137.3,163.4)
> m2=L_corr(m200, m800, null=0, exp.r=0.5, L.int=3, alpha=.05, verb=TRUE)

Support for observed correlation 0.6198 (dashed line) versus null of 0 (black line) = 5.776
 Support for specified correlation of 0.5 (blue line) versus observed r = -0.338
```

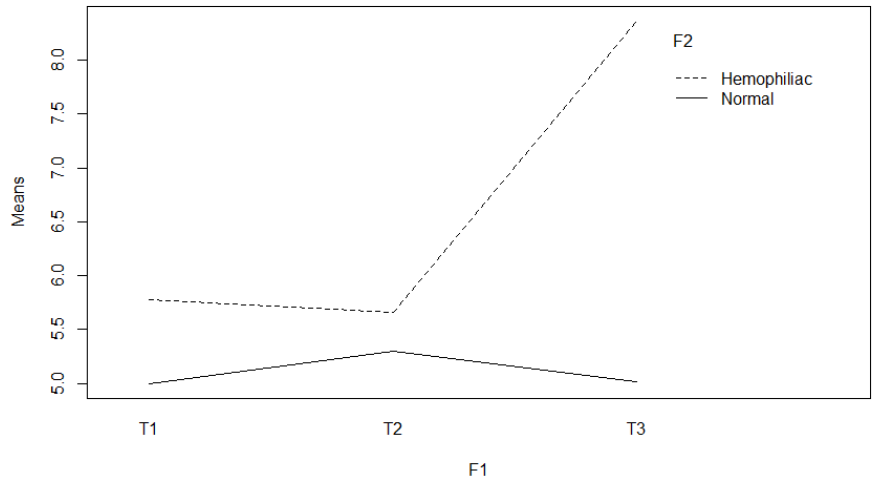


Figure 2: Graphical output from running the `L_2way_Factorial_ANOVA` function. The interaction plot shows the means for the 2 groups of patients (dashed line is hemophiliac, solid line is normal). The horizontal axis represents the 3 different treatments. An interaction is apparent from the plot, where the 3rd treatment shows a clear difference between the hemophiliac and normal patients.

```
Support for specified correlation versus null = 5.438
S-3 likelihood interval (red line) is from 0.19968 to 0.8474

P value = 0.00095
N = 25
```

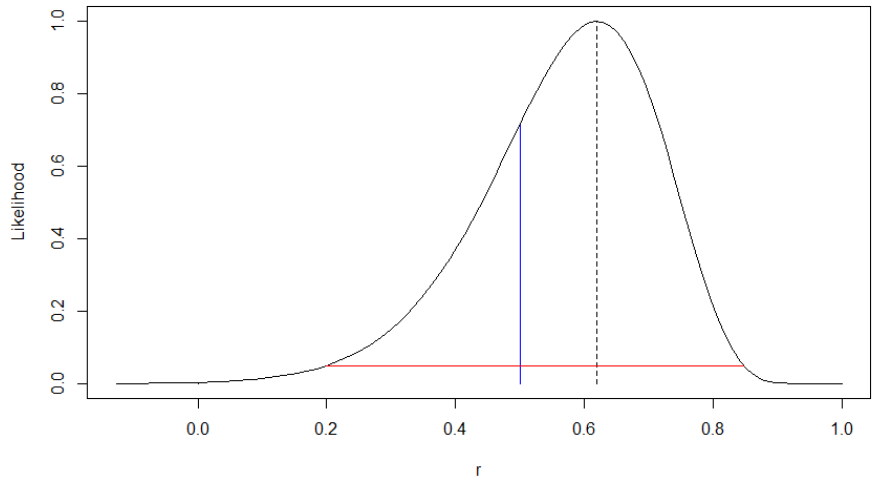


Figure 3: Graphical output from running the `L_corr` function. This is the likelihood function for the correlation coefficient, given the data. As before, the vertical dashed line is the MLE (the sample *r*), the blue line is a specified alternative value (0.5), and red horizontal line is the *S*-3 likelihood interval. The null value at 0 is outside of this interval and the evidence against it is extremely strong.

The assigned object m2 contains values of interest which can be assessed in the usual way with m2$..., but entering m2 on its own gives all the values.
The regression function `L_regress` only accommodates one predictor, while the logistic regression function `L_logistic_regress` allows up to 6 predictors, which need to be dummy coded for nominal data with more than 2 levels.

4 Categorical data

There are 5 different tests included in the package (excluding logistic regression mentioned above). The simplest is the one-way categorical data analysis using the function `L_1way_cat`. Two categories represents the binomial (giving the likelihood function plot), while multiple categories represents the multinomial distribution. The two-way categorical analysis uses the `L_2way_cat` function. For these two functions an additional evidence-based statistic S for the variance is calculated. This uses the formula given by Cahusac (2020b) p 158 and derived from Edwards (1992) p 187:

$$S = \frac{df}{2}\left(log\frac{df}{\chi^2_{df}}\right) - \frac{1}{2}(df - \chi^2_{df}), \tag{1}$$

where *df* is the degrees of freedom. This is most useful to test the variance in the model, specifically whether data are "too good to be true", i.e. the data fit a particular hypothesis closer than we would expect by chance (Edwards, 1986). Using just 2 categories in the one-way analysis can be demonstrated:

```
> obs <- c(18,5); exp.p <- c(0.7, 0.3)  # observed and expected values
> m3 <- L_1way_cat(obs, exp.p, verb = TRUE)

Binomial support for difference of MLE 0.7826087 (dashed line)
 from 0.7 (blue line) with 1 df = 0.398
 Support for variance differing more than expected = 0.019

 S-2 likelihood interval (red line) from 0.5853 to 0.91794

Chi-square(1) = 0.747,  p = 0.3872968
 Likelihood ratio test G(1) = 0.795, p = 0.37258, N = 23
 Likelihood-based 95% confidence interval from 0.58958 to 0.91597
```

As previously, the assigned object `m3` contains values of interest which can be assessed in the usual way with `m3$...`, but entering `m3` on its own gives all the values.

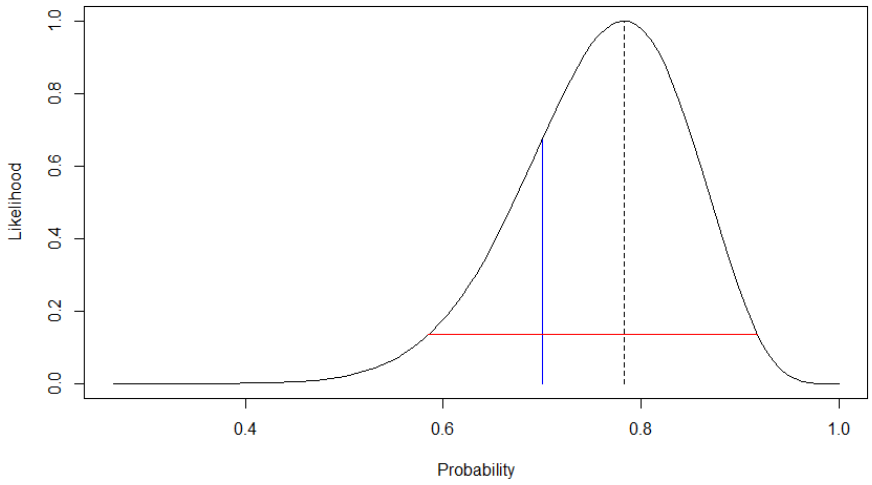


Figure 4: Graphical output from running the `L_1way_cat` function. This shows the likelihood function for the proportion, given the data. The vertical dashed line is the MLE, the blue line is the alternative hypothesis value (0.7), and the red horizontal line is the *S*-2 likelihood interval.

Some of the functions provide likelihood-based % confidence interval (Aitkin et al., 1989).
Finally, there are functions to calculate the odds ratio `L_OR`, relative risk `L_RR` and binomial efficacy `L_efficacy`.

5 Sample size calculations

The main challenge faced by the evidential researcher is of obtaining sufficiently strong evidence for or against one of two specified hypotheses. Like the probability of a Type II error, this probability is large with a small sample size and decreases as sample size increases. The function `L_t_test_sample_size` can be used to calculate the pre-study sample size for all the *t* tests (Cahusac and Mansour, 2022). The combined misleading and weak probability (Royall, 1997, 2000, 2004) is entered, with a default of 0.05, together with the strength of evidence desired (default = 3). For a paired samples test, where we wish to calculate sample size with a combined .2 probability of obtaining misleading or weak evidence, strength of evidence $S = 2$ and effect size 0.5, we would obtain a value of 38 by using the following:

```
> L_t_test_sample_size(MW = 0.2, sd = 1, d = 0.5, S = 2, paired = TRUE)

For 1 sample, or related samples, t test with M1 + W1 probability of 0.2
 Strength of evidence required is 2, and effect size of 0.5
 Required sample size = 38
```

The somewhat comparable calculation for Type II error of 0.2, two-sided alpha = 0.05 and same effect size of 0.5 produces a sample size of 34 (using stats::power.t.test(power = .80, delta = 0.5, sig.level=0.05, type="paired")).

6 Conclusions

The functions described for the likelihoodR package may be useful for those researchers and statisticians who wish to use the evidential approach for their data analysis (Cahusac, 2020a). In addition to the advantages mentioned earlier in the introduction there are other desirable features. First, categorical data analyses are not restricted by normality assumptions, and support values for independent components of cross-tabulated data sum precisely and algebraically (unlike such calculations in chi-square analyses). Second, in categorical and measurement analyses it is possible to show that the data fit the null (or other) hypothesis too well (e.g. for detection of data fraud). Third, analyses are versatile with unlimited complexity for model comparisons within a dataset, for example in ANOVA (Glover and Dixon, 2004).

As far as the author is aware, no other packages are available in R or other platforms. Currently users calculate likelihood ratios manually. This package addresses this shortcoming and hopefully will encourage more users to express their results in terms of log likelihood ratios.
One of the products of the likelihoodR package is that a module has been developed for jamovi (The jamovi project, 2021), named **jeva**, which includes many of these functions. Hopefully this will encourage further interest in the likelihood approach and facilitate teaching and research. The equivalent jamovi analysis is given earlier for the `L_ttest` produces output given in Figure 5. The output produced by jamovi is identical to that produced earlier by the package, although simpler in that it lacks the option of comparing an effect size (d, illustrated by the blue line in package output). The null versus observed (1st line of Support output) is -0.892, while in the package it is 0.892, the positive value being due to comparing observed versus the null. Other outputs match apart from decimal rounding, which can be selected in jamovi. The t, degrees of freedom (df) and p are the same for the null versus observed, although the jamovi output includes another line giving these statistics for the alternative hypothesis versus the observed. The jamovi output includes group descriptive statistics (although these are available from the assigned object in the package, e.g. `$obs.mean`. The jamovi output also includes the option of a descriptives plot (not shown) which displays the mean with specified likelihood interval.
The likelihoodR package described in this article provides a large number of functions, currently many more than envisaged for the jamovi module. As such, it will provide a major reference package for users interested in the likelihood approach.

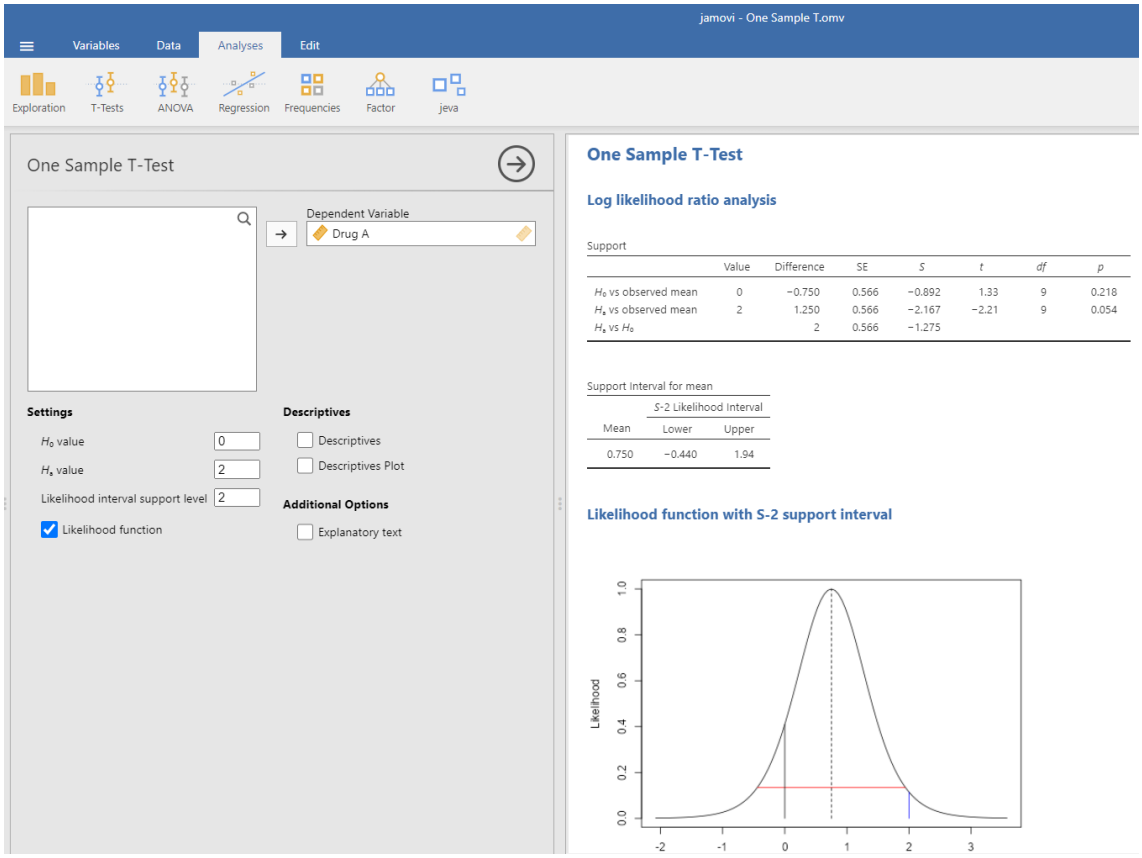


Figure 5: Graphical output from running the one-sample t test in the jamovi module **jeva** (created from the `L_ttest` function). The screenshot should be compared directly with Figure 1. On the left side is the dialog box where a variable can be selected (Drug A), and various settings chosen, including the null and alternative hypothesis values, and to display the likelihood function. An additional option gives an explanation about the obtained support values and likelihood interval, and their interpretation. On the right side is a summary of the analysis, at the top showing the S values for hypothesis comparisons (see text for the interpretation of the tabulated values). Below this is shown the support interval, and finally below that the likelihood function. The line colours for the likelihood function are the same as those given in Figure 1, although there is no option for a specified effect size (no green line).

Bibliography

M. Aitkin, D. Anderson, B. Francis, and J. Hinde. *Statistical modelling in GLIM*. Oxford Statistical Science Series, Oxford, UK, 1989. ISBN 978-0198522041. URL https://www.amazon.co.uk/Statistical-Modelling-GLIM-Oxford-Science/dp/0198522045/. [p209]

P. Cahusac. Data as evidence. *Experimental Physiology*, 105(7):1071–1080, 2020a. ISSN 0958-0670. doi: 10.1113/ep088664. URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/EP088664. [p209]

P. M. B. Cahusac. *Evidence-Based Statistics: An Introduction to the Evidential Approach – from Likelihood Principle to Statistical Practice*. John Wiley and Sons, New Jersey, 2020b. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9781119549833. [p203, 204, 206, 208]

P. M. B. Cahusac. *Package 'likelihoodR'*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://cran.r-project.org/web/packages/likelihoodR/index.html. [p203]

P. M. B. Cahusac and S. E. Mansour. Estimating sample sizes for evidential t tests. *Research in Mathematics*, 9(1):1–12, 2022. ISSN 2768-4830. doi: 10.1080/27684830.2022.2089373. URL https://www.tandfonline.com/doi/full/10.1080/27684830.2022.2089373. [p209]

Z. Dienes. *Understanding Psychology as a Science: An introduction to scientific and statistical inference*. Palgrave MacMillan, 2008. ISBN 978-0-230-54230-3. URL https://www.amazon.co.uk/Understanding-Psychology-Science-Introduction-Statistical/dp/023054231X. [p203]

P. Dixon. The p-value fallacy and how to avoid it. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie experimentale*, 57(3):189–202, 2003. ISSN 1878-7290(Electronic),1196-1961(Print). doi: 10.1037/h0087425. URL https://pubmed.ncbi.nlm.nih.gov/14596477/. [p203, 206]

P. Dixon. The effective number of parameters in post hoc models. *Behavior research methods*, 45(3): 604–612, 2013. ISSN 1554-3528. URL https://link.springer.com/article/10.3758/s13428-013-0373-7. [p206]

A. W. F. Edwards. Are mendel's results really too close? *Biological Reviews*, 61(4):295–312, 1986. ISSN 1464-7931. doi: 10.1111/j.1469-185X.1986.tb00656.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-185X.1986.tb00656.x. [p208]

A. W. F. Edwards. *Likelihood*, volume 2nd edition. John Hopkins University Press, Baltimore, 1992. ISBN 0801844436. URL https://www.amazon.co.uk/Likelihood-Dr-W-F-Edwards/dp/0801844452. [p203, 208]

S. Glover and P. Dixon. Likelihood ratios: A simple and flexible statistic for empirical psychologists. *Psychonomic bulletin and review*, 11(5):791–806, 2004. ISSN 1069-9384. doi: 10.3758/BF03196706. URL https://link.springer.com/article/10.3758/BF03196706. [p206, 209]

S. N. Goodman. Toward evidence-based medical statistics. 1: The p value fallacy. *Annals of Internal Medicine*, 130(12):995–1004, 1999. ISSN 0003-4819. doi: 10.7326/0003-4819-130-12-199906150-00008. URL http://dx.doi.org/10.7326/0003-4819-130-12-199906150-00008. [p203]

S. N. Goodman and R. M. Royall. Evidence and scientific research. *American Journal of Public Health*, 78(12):1568–1574, 1988. ISSN 0090-0036. doi: 10.2105/AJPH.78.12.1568. URL https://ajph.aphapublications.org/doi/10.2105/AJPH.78.12.1568. [p203, 204]

S. Greenland. Valid p-values behave exactly as they should: Some misleading criticisms of p-values and their resolution with s-values. *The American Statistician*, 73(1):106–114, 2019. ISSN 0003-1305. doi: 10.1080/00031305.2018.1529625. URL https://cogentoa.tandfonline.com/doi/full/10.1080/00031305.2018.1529625. [p204]

C. M. Hurvich and C.-L. Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989. ISSN 0006-3444. doi: 10.1093/biomet/76.2.297. URL https://doi.org/10.1093/biomet/76.2.297. [p206]

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. URL https://doi.org/10.1080/10618600.1996.10474713. [p203]

H. Jeffreys. Further significance tests. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 32, pages 416–445. Cambridge University Press, 1936. ISBN 1469-8064. URL https://www.cambridge.org/core/journals/mathematical-proceedings-of-the-cambridge-philosophical-society/article/abs/further-significance-tests/B2FA761E09518833D0F8FE96B715F932. [p203]

D. Lakens. Why p-values should be interpreted as p-values and not as measures of evidence. 2021. URL https://daniellakens.blogspot.com/2021/11/why-p-values-should-be-interpreted-as-p.html. [p203]

G. Palm. *Novelty, Information and Surprise*. Springer Berlin, Heidelberg, 2012. ISBN 978-3-642-43583-6. doi: 10.1007/978-3-642-29075-6. URL https://link.springer.com/book/10.1007/978-3-642-29075-6. [p204]

R. Royall. *The Likelihood paradigm for statistical evidence*. University of Chicago, Chicago, 2004. URL https://press.uchicago.edu/ucp/books/book/chicago/N/bo3635206.html. [p209]

R. M. Royall. *Statistical Evidence: a Likelihood Paradigm*. Monographs on statistics and applied probability. Chapman and Hall, London, 1997. ISBN 0-412-04411-0. URL https://www.routledge.com/Statistical-Evidence-A-Likelihood-Paradigm/Royall/p/book/9780412044113. [p203, 209]

R. M. Royall. On the probability of observing misleading statistical evidence. *Journal of the American Statistical Association*, 95(451):760–768, 2000. ISSN 0162-1459. doi: 10.1080/01621459.2000.10474264. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.2000.10474264. [p209]

The jamovi project. *The jamovi project*. jamovi (Version 2.3.18.0) [Computer Software], Sydney, Australia, 2021. URL https://www.jamovi.org. [p210]

R. L. Wasserstein and N. A. Lazar. The ASA's statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016. ISSN 0003-1305. doi: 10.1080/00031305.2016.1154108. URL https://doi.org/10.1080/00031305.2016.1154108. [p203]

Peter Cahusac
College of Medicine, Alfaisal University, Riyadh
Department of Pharmacology & Biostatistics
And Department of Comparative Medicine
King Faisal Specialist Hospital & Research Centre, Riyadh
Kingdom of Saudi Arabia
ORCiD 0000-0003-4976-2834
pcahusac@alfaisal.edu

CIMTx: An R Package for Causal Inference with Multiple Treatments using Observational Data

by Lianyuan Hu and Jiayi Ji

Abstract CIMTx provides efficient and unified functions to implement modern methods for causal inferences with multiple treatments using observational data with a focus on binary outcomes. The methods include regression adjustment, inverse probability of treatment weighting, Bayesian additive regression trees, regression adjustment with multivariate spline of the generalized propensity score, vector matching and targeted maximum likelihood estimation. In addition, CIMTx illustrates ways in which users can simulate data adhering to the complex data structures in the multiple treatment setting. Furthermore, the CIMTx package offers a unique set of features to address the key causal assumptions: positivity and ignorability. For the positivity assumption, CIMTx demonstrates techniques to identify the common support region for retaining inferential units using inverse probability of treatment weighting, Bayesian additive regression trees and vector matching. To handle the ignorability assumption, CIMTx provides a flexible Monte Carlo sensitivity analysis approach to evaluate how causal conclusions would be altered in response to different magnitude of departure from ignorable treatment assignment.

1 Introduction

Modern comparative effectiveness research (CER) questions often require comparing the effectiveness of multiple treatments on a binary outcome (Hu et al., 2020a). To answer these CER questions, specialized causal inference methods are needed. Methods appropriate for drawing causal inferences about multiple treatments include regression adjustment (RA) (Rubin, 1973; Linden et al., 2016), inverse probability of treatment weighting (IPTW) (Feng et al., 2012; McCaffrey et al., 2013), Bayesian Additive Regression Trees (BART) (Hill, 2011; Hu et al., 2021b, 2020a), regression adjustment with multivariate spline of the generalized propensity score (RAMS) (Hu and Gu, 2021), vector matching (VM) (Lopez and Gutman, 2017) and targeted maximum likelihood estimation (TMLE) (Rose and Normand, 2019). Drawing causal inferences using observational data, however, inevitably requires assumptions. A key causal identification assumption is the *positivity* or sufficient overlap assumption, which implies that there are no values of pre-treatment covariates that could occur only among units receiving one of the treatments (Hu et al., 2020a). Another key assumption requires appropriately conditioning on all pre-treatment variables that predict both treatment and outcome. The pre-treatment variables are known as confounders and this requirement is referred to as the *ignorability* assumption (also as no unmeasured confounding) (Hu et al., 2022b). An important strategy to handle the positivity assumption is to identify a common support region for retaining inferential units. The ignorability assumption can be violated in observational studies, and as a result can lead to biased treatment effect estimates. One widely recognized way to address such concerns is sensitivity analysis (Erik von Elm et al., 2007; Hu et al., 2022b).

The CIMTx package provides a suite of functions to easily implement the causal estimation methods, many of which were recently developed (Lopez and Gutman, 2017; Hu et al., 2020a; Hu and Gu, 2021). In addition, **CIMTx** provides strategies to define a common support region to address the positivity assumption using IPTW, BART, VM and implements a flexible Monte Carlo sensitivity analysis approach (Hu et al., 2022b) for unmeasured confounding to address the ignorability assumption. Finally, **CIMTx** offers detailed examples of how to simulate data adhering to the complex structures in the multiple treatment setting. The simulated data can then be used by an analyst to compare the performance of different causal estimation methods. Table 1 summarizes key functionalities of **CIMTx** in comparison to recent R packages designed for causal inference with multiple treatments using observational data. **CIMTx** provides a comprehensive set of functionalities: from simulating data to estimating the causal effects to addressing causal assumptions and elucidating their ramifications. To assist applied researchers and practitioners who work with observational data and wish to draw inferences about the effects of multiple treatments, this article provides a comprehensive illustration of the **CIMTx** package.

Table 1: Comparisons of R packages for causal inference.

R packages	Continuous Outcome	Binary Outcome	Sensitivity Analysis	Identification of Common Support	Design factors	Estimation procedure
CIMTx	×	✔	✔	✔*	✔	RA, IPTW-SL IPTW-Multinomial IPTW-GBM VM, BART RAMS, TMLE
PSweight	✔	✔	×	✔	×	OW, IPTW-SL IPTW-Multinomial IPTW-GBM
twang	✔	×	✔	×	×	IPTW-GBM
WeightIt	✔	×	×	✔	×	CBPS, IPTW-SL IPTW-Multinomial IPTW-GBM,EBCW IPTW-TSBW
CBPS	✔	✔	×	×	×	CBPS
optweight	✔	×	×	×	×	IPTW-TSBW

✔: the feature is offered in the method; × indicates otherwise; RA: Regression adjustment; IPTW: Inverse probability of treatment weighting; BART: Bayesian additive regression trees; RAMS: Regression adjustment with multivariate spline of generalized propensity score; VM: Vector matching; TMLE: Targeted maximum likelihood estimation; CBPS: Covariate balancing propensity score; OW: Overlap weights; IPTW-Multinomial: Inverse probability of treatment weighting with weight estimated by multinomial logistic regression; IPTW-GBM: Inverse probability of treatment weighting with weight estimated by generalized boosted model; IPTW-SL: Inverse probability of treatment weighting with weight estimated by super learner; IPTW-TSBW: Inverse probability of treatment weighting with targeted stable balancing weights; EBCW: Empirical balancing calibration weights.
*: Identification of Common Support is only for VM, BART and IPTW related methods
References: **PSweight** (Version 1.1.4): Zhou et al. (2020);**twang** (Version 1.6) Ridgeway et al. (2020);**WeightIt** (Version 0.10.2) Greifer (2020);**CBPS** (Version 0.22): Fong et al. (2021); **optweight** (Version 0.2.5): Greifer (2019);

2 Design factors for data simulation

CIMTx provides specific functions to simulate data possessing complex data characteristics of the multiple treatment setting. Seven design factors are considered: (1) sample size, (2) ratio of units across treatment groups, (3) whether the treatment assignment model and the outcome generating model are linear or nonlinear, (4) whether the covariates that best predict the treatment also predict the outcome well, (5) whether the response surfaces are parallel across treatment groups, (6) outcome prevalence, and (7) degree of covariate overlap.

Design factors (1)–(5)

For the data generating process of treatment assignment, consider a multinomial logistic regression model,

$$\begin{aligned} \ln \frac{P(W=1)}{P(W=T)} &= \delta_1 + \boldsymbol{X}\xi_1^L + \boldsymbol{Q}\xi_1^{NL} \\ &\vdots \\ \ln \frac{P(W=T-1)}{P(W=T)} &= \delta_{(T-1)} + \boldsymbol{X}\xi_{(T-1)}^L + \boldsymbol{Q}\xi_{(T-1)}^{NL}, \end{aligned} \tag{1}$$

where $\boldsymbol{Q}$ denotes the nonlinear transformations and higher-order terms of the predictors $\boldsymbol{X}$. $\xi_1^L, \ldots, \xi_{(T-1)}^L$ are vectors of coefficients for the untransformed versions of the predictors $\boldsymbol{X}$ and $\xi_1^{NL}, \ldots, \xi_{(T-1)}^{NL}$ for the transformed versions of the predictors captured in $\boldsymbol{Q}$. The intercepts $\delta_1, \ldots, \delta_{(T-1)}$ can be specified to create the corresponding ratio of units across T treatment groups. The T sets of potential response

surfaces can be generated as follows:

$$\begin{aligned} E[Y(1)|\boldsymbol{X}] &= \text{logit}^{-1}\{\tau_1 + \boldsymbol{X}\gamma_1^L + \boldsymbol{Q}\gamma_1^{NL}\} \\ &\vdots \\ E[Y(T)|\boldsymbol{X}] &= \text{logit}^{-1}\{\tau_T + \boldsymbol{X}\gamma_T^L + \boldsymbol{Q}\gamma_T^{NL}\}, \end{aligned} \tag{2}$$

where the coefficient setting $\gamma_1^L = \ldots = \gamma_T^L$, $\gamma_1^{NL} = \ldots = \gamma_T^{NL}$ and $\tau_1 \neq \ldots \neq \tau_T$ corresponds to the parallel response surfaces, and by assigning different values to γ_w^L and γ_w^{NL} and setting $\tau_1 = \ldots = \tau_T = 0$, nonparallel response surfaces are generated, which imply treatment effect heterogeneity. Note that the predictors $\boldsymbol{X}$ and the transformed versions of the predictors $\boldsymbol{Q}$ in the treatment assignment model (1) can be different than those in the outcome generating model (2) to create various degrees of alignment. The observed outcomes are related to the potential outcomes through $Y_i = \sum_{w_i \in \mathscr{W}} Y_i(w)$. Covariates $\boldsymbol{X}$ can be generated from user-specified data distributions.

Outcome prevalence

Values for parameters $\tau_1, \ldots, \tau_T$ in model (2) can be chosen to create various outcome prevalence rates. The outcomes are considered rare if the prevalence rate is $< 5\%$.

Covariate overlap

With observational data, it is important to investigate how the sparsity of covariate overlap impacts the estimation of causal effects. We can modify the formulation of the treatment assignment model (1) to adjust the sparsity of overlap by including a multiplier parameter ψ (Hu et al., 2021a) as follows:

$$\begin{aligned} \ln \frac{P(W=1)}{P(W=T)} &= \delta_1 + \boldsymbol{X}\psi\xi_1^L + \boldsymbol{Q}\psi\xi_1^{NL} \\ &\vdots \\ \ln \frac{P(W=T-1)}{P(W=T)} &= \delta_{(T-1)} + \boldsymbol{X}\psi\xi_{(T-1)}^{NL} + \boldsymbol{Q}\psi\xi_{(T-1)}^{NL}, \end{aligned} \tag{3}$$

where larger values of ψ correspond to increased sparsity degrees of overlap.

Implementation in CIMTx

We will first demonstrate the functionality of `data_sim()` in **CIMTx** to simulate data in the multiple treatment setting using the above 7 design factors. We first use the `data_sim()` function to simulate a dataset with the following characteristics: (1) sample size = 500, (2) ratio of units = 1:1:1 across three treatment groups, (3) nonlinear treatment assignment and outcome generating models, (4) different predictors for the treatment assignment and outcome generating mechanisms, (5) parallel response surfaces, (6) outcome prevalence = $(0.16, 0.51, 0.75)$ in three treatment groups with an overall rate around 0.5 and (7) moderate covariate overlap. Note that for the design factor (6), we can adjust `tau` to generate rare outcome events.

The outputs of the simulated data object are: (1) `data$covariates` for $\boldsymbol{X}$, (2) `data$w` for treatment indicators, (3) `data$y` for observed binary outcomes, (4) `data$y_prev` for the outcome prevalence rates, (5) `data$ratio_of_units` for the proportions of units in each treatment group, (6) `data$overlap_fig` for the visualization of covariate overlap via boxplots of the distributions of true generalized propensity score (GPS).

```
library(CIMTx)
set.seed(1)
data <- data_sim(
  sample_size = 500, n_trt = 3,
  x = c("rnorm(0, 0.5)",  # x1
        "rbeta(2, .4)",   # x2
        "runif(0, 0.5)",  # x3
        "rweibull(1, 2)", # x4
        "rbinom(1, .4)"),   # x5
  # linear terms in parallel response surfaces
  lp_y = rep(".2*x1 + .3*x2 - .1*x3 - .1*x4 - .2*x5", 3),
```

```
  # nonlinear terms in parallel response surfaces
  nlp_y  = rep(".7*x1*x1  - .1*x2*x3", 3),
  align = F,# different predictors used in treatment and outcome models
  # linear terms in treatment assignment model
  lp_w = c(".4*x1 + .1*x2  - .1*x4 + .1*x5",   # w = 1
           ".2*x1 + .2*x2  - .2*x4 - .3*x5"),  # w = 2
  # nonlinear terms in treatment assignment model
  nlp_w = c("-.5*x1*x4  - .1*x2*x5", # w = 1
            "-.3*x1*x4 + .2*x2*x5"), # w = 2
  tau = c(-1.5, 0, 1.5), delta = c(0.5, 0.5), psi = 1)
```

In this simulated dataset, the ratio of units (`data$ratio_of_units`) and outcome prevalences (`data$y_prev`) are:

```
#> w
#>    1    2    3
#> 0.35 0.35 0.30

#>         w y_prev
#> 1       1   0.16
#> 2       2   0.51
#> 3       3   0.75
#> 4 Overall   0.46
```

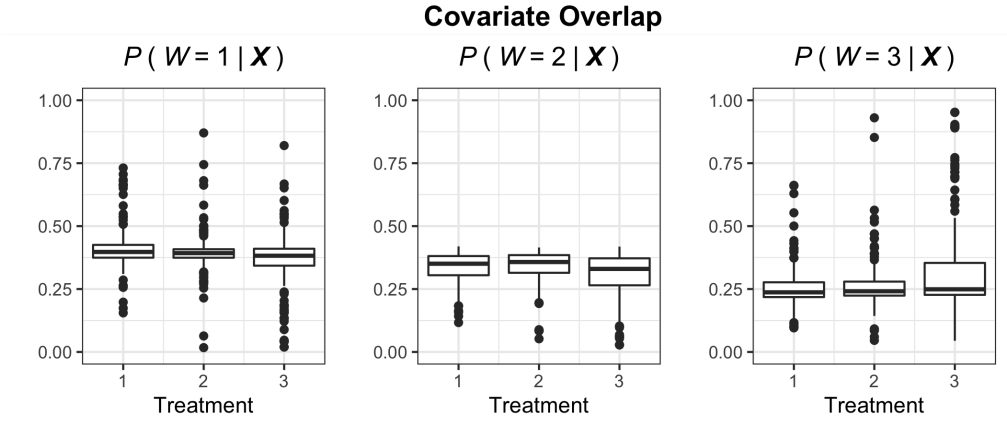


Figure 1: Moderate overlap with `psi = 1`. Each panel presents boxplots by treatment group of the true generalized propensity score for one of the treatments, $P(W_i = w \mid \boldsymbol{X} = \boldsymbol{x})$ for every unit in the sample. The left-hand panel presents treatment 1 ($W = 1$), the middle panel presents treatment 2 ($W = 2$), and the right-hand panel presents treatment 3 ($W = 3$).

Figure 1 (`data$overlap_fig`) shows the distributions of true GPS for each treatment group, suggesting moderate covariate overlap. We can change structures of the simulated data by modifying arguments of the `data_sim()` function. For example, setting `delta = c(1.5,0.5)` yields unequal sample sizes across treatment groups with the ratio of unit .6 : .2 : .2. Assigning smaller values to `psi` can increase overlap: `psi = 0.1` corresponds to a strong covariate overlap as shown in Figure 2.

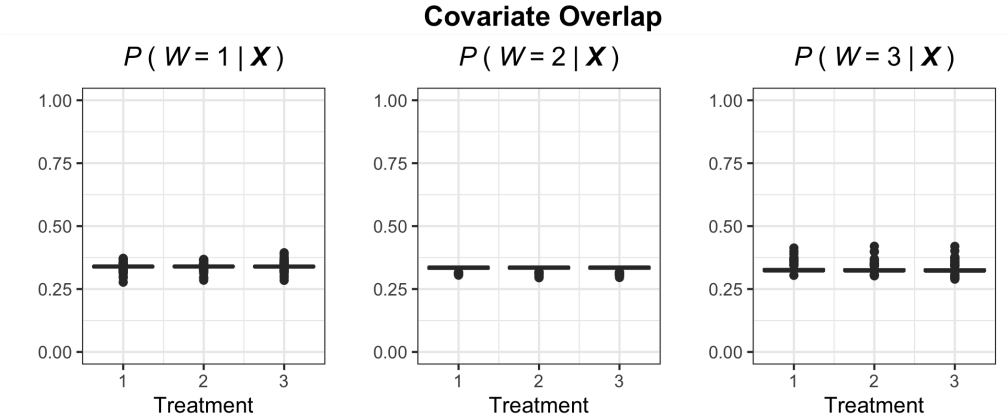


Figure 2: Strong overlap with `psi = 0.1`. Each panel presents boxplots by treatment group of the true generalized propensity score for one of the treatments for every unit in the sample.

3 Methodology and implementation in CIMTx

Estimation of causal effects

Consider an observational study with N individuals, indexed by $i = 1, \ldots, N$, drawn randomly from a target population. Each individual was exposed to one and only one treatment, indexed by W. The goal of this study is to estimate the causal effect of treatment W on a binary outcome Y. There are a total of T possible treatments, and $W_i = w$ if individual i is observed under treatment w, where $w \in \mathscr{W} = \{1, 2, \ldots, T\}$. Pre-treatment measured confounders are indexed by $\boldsymbol{X}_i$. Under the potential outcomes framework, (Rubin, 1974; Holland, 1986), individual i has T potential outcomes $\{Y_i(1), \ldots, Y_i(T)\}$ under each treatment of $\mathscr{W}$. For each individual, at most one of the potential outcomes is observed – the one corresponding to the treatment to which the individual is exposed. All other potential outcomes are missing, which is known as the fundamental problem of causal inference (Holland, 1986). In general, three standard causal identification assumptions (Rubin, 1980; Hu et al., 2020a) need to be maintained in order to estimate the causal effects from observational data:

(A1) The stable unit treatment value assumption: there is no interference between units and there are no different versions of a treatment.

(A2) Positivity: the GPS for treatment assignment $e(\boldsymbol{X}_i) = P(W_i = 1 \mid \boldsymbol{X}_i)$ is bounded away from 0 and 1.

(A3) Ignorability: pre-treatment covariates $\boldsymbol{X}_i$ are sufficiently predictive of both treatment assignment and outcome, $p(W_i \mid Y_i(1), \ldots, Y_i(T), \boldsymbol{X}_i) = p(W_i \mid \boldsymbol{X}_i)$.

The **CIMTx** package addresses assumption (A2) in the section of "Identification of a common support region" and (A3) in the section of "Sensitivity analysis for unmeasured confounding".

Causal effects can be estimated by summarizing functionals of individual-level potential outcomes. For dichotomous outcomes, causal estimands can be the risk difference (RD), odds ratio (OR) or relative risk (RR). For purposes of illustration, we define causal effects based on the RD. Let s_1 and s_2 be two subgroups of treatments such that $s_1, s_2 \subset \mathscr{W}$ and $s_1 \cap s_2 = \emptyset$, and define $|s_1|$ as the cardinality of s_1 and $|s_2|$ of s_2. Two commonly used causal estimands are the average treatment effect (ATE), ATE_{s_1,s_2}, and the average treatment effect on the treated (ATT), for example, among those receiving s_1, $ATT_{s_1|s_1,s_2}$. They are defined as:

$$\begin{aligned} ATE_{s_1,s_2} &= E\left[\frac{\sum_{w \in s_1} Y_i(w)}{|s_1|} - \frac{\sum_{w' \in s_2} Y_i(w')}{|s_2|}\right], \\ ATT_{s_1|s_1,s_2} &= E\left[\frac{\sum_{w \in s_1} Y_i(w)}{|s_1|} - \frac{\sum_{w' \in s_2} Y_i(w')}{|s_2|} \middle| W_i \in s_1\right]. \end{aligned} \tag{4}$$

We now introduce six methods implemented in **CIMTx** for estimating the causal effects of multiple treatments: RA, IPTW, BART, RAMS, VM and TMLE.

Regression adjustment Regression adjustment (Rubin, 1973; Linden et al., 2016), also known as model-based imputation (Imbens and Rubin, 2015), uses a regression model to impute missing

potential outcomes: what would have happened to a specific individual had this individual received a treatment to which he or she not exposed. RA regresses the outcomes on treatment and confounders,

$$f(w, \boldsymbol{X}_i) = E[Y_i \mid W_i = w, \boldsymbol{X}_i] = \text{logit}^{-1}\left\{\beta_0 + \beta_1 w + \boldsymbol{\beta}_2^\top \boldsymbol{X}_i\right\}, \tag{5}$$

where β_0 is the intercept, β_1 is the coefficient for treatment and $\boldsymbol{\beta}_2$ is a vector of coefficients for covariates $\boldsymbol{X}_i$. From the fitted regression model (5), the missing potential outcomes for each individual are imputed using the observed data. The causal effects can be estimated by contrasting the imputed potential outcomes between treatment groups. **CIMTx** implements RA with the Bayesian logistic regression model via the `bayesglm()` function of the **arm** package. For the ATE effects, we first average the L predictive posterior draws $\{f^l(w, \boldsymbol{X}_i), l = 1, \ldots, L\}$ over the empirical distribution of $\{\boldsymbol{X}_i\}_{i=1}^N$, and for the ATT effects using s_1 as the reference group, over the empirical distribution of $\{\boldsymbol{X}_i\}_{i:W_i \in s_1}$. We then take the difference of the averaged values between two treatment groups $w \in s_1$ and $w' \in s_2$. Inferences about treatment effect can be obtained based on the L posterior average treatment effects. The 95% credible interval is calculated using the 2.5th percentile and the 97.5th percentile of the posterior draws (Kruschke, 2014).

In our package **CIMTx**, we can specify `method = "RA"` and `estimand = "ATE"` in the `ce_estimate()` function to get the ATE effects via RA:

```
ra_ate_res <- ce_estimate(y = data$y, x = data$covariates, w = data$w,
                          method = "RA", estimand = "ATE", ndpost = 100)
```

The estimates, standard errors and 95% confidence intervals for the causal estimands would be printed using the `summary()` generic function:

```
summary(ra_ate_res)
```

```
#> $ATE12
#>      EST   SE LOWER UPPER
#> RD -0.28 0.04 -0.35 -0.21
#> RR  0.39 0.07  0.27  0.54
#> OR  0.26 0.06  0.17  0.40

#> $ATE13
#>      EST   SE LOWER UPPER
#> RD -0.60 0.06 -0.69 -0.47
#> RR  0.23 0.05  0.15  0.34
#> OR  0.07 0.02  0.03  0.13

#> $ATE23
#>      EST   SE LOWER UPPER
#> RD -0.31 0.04 -0.39 -0.25
#> RR  0.60 0.04  0.52  0.67
#> OR  0.25 0.05  0.17  0.34
```

Specifying `estimand = "ATT"` and setting `reference_trt` will get us the ATT effects:

```
ra_att_res <- ce_estimate(y = data$y, x = data$covariates,w = data$w, method = "RA",
                          estimand = "ATT", ndpost = 100, reference_trt = 1)
summary(ra_att_res)
```

```
#> $ATT12
#>      EST   SE LOWER UPPER
#> RD -0.28 0.05 -0.37 -0.18
#> RR  0.40 0.09  0.25  0.57
#> OR  0.27 0.08  0.16  0.44

#> $ATT13
#>      EST   SE LOWER UPPER
#> RD -0.59 0.06 -0.67 -0.46
#> RR  0.24 0.06  0.14  0.38
#> OR  0.07 0.03  0.03  0.13
```

Inverse probability of treatment weighting The idea of IPTW was originally introduced by Horvitz and Thompson (1952) in survey research to adjust for imbalances in sampling pools. Weighting

methods have been extended to estimate the causal effect of a binary treatment in observational studies, and more recently reformulated to accommodate multiple treatments (Imbens, 2000; Feng et al., 2012; McCaffrey et al., 2013). When interest is in estimating the pairwise ATE for treatment groups s_1 and s_2, a consistent estimator of ATE_{s_1,s_2} is given by the weighted mean,

$$\widehat{ATE}_{s_1,s_2} = \frac{\sum_{i=1}^{N} Y_i I(W_i \in s_1)/|s_1|}{\sum_{i=1}^{N} I(W_i \in s_1) r(W_i, \boldsymbol{X_i})} - \frac{\sum_{i=1}^{N} Y_i I(W_i \in s_2)/|s_2|}{\sum_{i=1}^{N} I(W_i \in s_2) r(W_i, \boldsymbol{X_i})}, \tag{6}$$

where $r(w, \boldsymbol{X}_i)$ is the weights satisfying $r(w, \boldsymbol{X}_i) = 1/P(W_i = w \mid \boldsymbol{X}_i)$, and $I(\cdot)$ is the indicator function. The **CIMTx** package provides three ways in which the weights can be estimated: (i) multinomial logistic regression (Feng et al., 2012), (ii) generalized boosted model (GBM) (McCaffrey et al., 2013), and (iii) super learner (Van der Laan et al., 2007). A challenge with IPTW is low GPS can result in extreme weights, which may yield erratic causal estimates with large sample variances (Little, 1988; Kang et al., 2007). This issue is increasingly likely as the number of treatments increases. Weight trimming or truncation can alleviate the issue of extreme weights (Cole and Hernán, 2008; Lee et al., 2011)). **CIMTx** provides an argument for users to choose the percentile at which the weights should be truncated. We briefly describe the three weight estimators.

(i) The multinomial logistic regression model for treatment assignment is as follows:

$$P(W_i = w|\boldsymbol{X}_i) = \frac{e^{\boldsymbol{\alpha}_w' \boldsymbol{X}_i}}{1 + e^{\boldsymbol{\alpha}_1' \boldsymbol{X}_i} + \ldots + e^{\boldsymbol{\alpha}_{T-1}' \boldsymbol{X}_i}},$$

where $\boldsymbol{\alpha}_w'$ is a vector of coefficients for $\boldsymbol{X}_i$ corresponding to treatment w, and can be estimated by using an iterative procedure such as generalized iterative scaling or iteratively reweighted least squares.

(ii) GBM uses machine learning to flexibly model the relationships between treatment assignment and covariates. It does this by growing a series of boosted classification trees to minimize an exponential loss function. This process is effective for fitting nonlinear treatment models characterized by curves and interactions. The procedure of estimating the GPS can be tuned to find the GPS model producing the best covariate balance between treatment groups.

(iii) Super learner is an algorithm that creates the optimally weighted average of several machine learning models. The machine learning models can be specified via the `SL.library` argument of the **SuperLearner** package. This approach has been proven to be asymptotically as accurate as the best possible prediction algorithm that is included in the library (Van der Laan et al., 2007).

IPTW can be implemented in **CIMTx** by setting a specific method and estimand. For IPTW estimators, variance can be estimated via a robust sandwich-type variance estimator or a bootstrap variance estimator. In practice, a bootstrap variance estimator is often recommended. (Austin, 2016). The following shows the code to estimate ATE using IPTW with weights estimated by multinomial logistic regression.

```
iptw_multi_res <- ce_estimate(y = data$y, x = data$covariates , w = data$w,
                              method = "IPTW-Multinomial", estimand = "ATE")
```

We can estimate the ATE effects with weights estimated by super learner and GBM by changing the argument of `method` to `"IPTW-SL"`,`"IPTW-GBM"` respectively. We can then estimate the causal effects and bootstrap confidence intervals by setting `boot = TRUE`.

```
iptw_sl_trim_ate_res <- ce_estimate(y = data$y, x = data$covariates , w = data$w,
                                    method = "IPTW-SL", estimand = "ATE",
                                    sl_library =  c("SL.glm", "SL.glmnet", "SL.rpart"),
                                    trim_perc = c(0.05,0.95), boot = TRUE,
                                    nboots = 100, verbose_boot = F)

summary(iptw_sl_trim_ate_res)
#> $ATE12
#>      EST   SE LOWER UPPER
#> RD -0.34 0.05 -0.42 -0.24
#> RR  0.33 0.07  0.19  0.48
#> OR  0.20 0.06  0.10  0.33

#> $ATE13
#>      EST   SE LOWER UPPER
#> RD -0.59 0.05 -0.67 -0.46
```

```
#> RR  0.22 0.05  0.13  0.34
#> OR  0.07 0.02  0.04  0.13

#> $ATE23
#>      EST   SE LOWER UPPER
#> RD -0.25 0.05 -0.34 -0.15
#> RR  0.67 0.06  0.57  0.79
#> OR  0.34 0.09  0.21  0.54
```

Bayesian additive regression trees BART (Chipman et al., 2010) is a likelihood-based machine learning model and has been adapted into causal inference settings in recent years (Hill, 2011; Hu et al., 2020a; Hu and Gu, 2021; Hu et al., 2021a,c). For a binary outcome, BART uses the probit regression

$$f(w, \boldsymbol{X}_i) = E[Y_i|W_i = w, \boldsymbol{X}_i] = \Phi\Big\{ \sum_{j=1}^{J} g_j(w, \boldsymbol{X}_i; T_j, M_j) \Big\}, \tag{7}$$

where Φ is the the standard normal cumulative distribution function, (T_j, M_j) indexes a single subtree model in which T_j denotes the regression tree and M_j is a set of parameter values associated with the terminal nodes of the jth regression tree, $g_j(w, \boldsymbol{X}_i; T_j, M_j)$ represents the mean assigned to the node in the jth regression tree associated with covariate value $\boldsymbol{X}_i$ and treatment level w, and the number of regression trees J is considered to be fixed and known. BART uses regularizing priors for (T_j, M_j) to keep the impact of each tree small. Although the prior distributions can be specified via the `ce_estimate()` function of **CIMTx**, the default priors tend to work well and require little modification in many situations (Hill, 2011; Hu et al., 2020a,b). The details of prior specification and Bayesian backfitting algorithm for posterior sampling can be found in Chipman et al. (2010). The posterior inferences about the treatment effects can be drawn in a similar way as described in the Regression adjustment section.

Setting `method = "BART"` and specifiying the `estimand = "ATE"` or `estimand = "ATT"` of the `ce_estimate()` function implements the BART method.

```
bart_res <- ce_estimate(y = data$y, x = data$covariates, w = data$w, method = "BART",
                        estimand = "ATT", ndpost=100, reference_trt = 1)

summary(bart_res)
```

```
#> $ATT12
#>      EST   SE LOWER UPPER
#> RD -0.38 0.07 -0.51 -0.25
#> RR  0.47 0.08  0.31  0.61
#> OR  0.21 0.07  0.10  0.35

#> $ATT13
#>      EST   SE LOWER UPPER
#> RD -0.56 0.07 -0.69 -0.43
#> RR  0.38 0.07  0.24  0.50
#> OR  0.06 0.03  0.02  0.13
```

Regression adjustment with multivariate spline of GPS For a binary outcome, the number of outcome events can be small. The estimation of causal effects is challenging with rare outcomes because the great majority of units contribute no information to explaining the variability attributable to the differential treatment regimens in the health outcomes (Hu and Gu, 2021). Franklin et al. (2017) found that regression adjustment on propensity score using one nonlinear spline performed best with respect to bias and root-mean-squared-error in estimating treatment effects. Hu and Gu (2021) proposed RAMS, which accommodates multiple treatments by using a nonlinear spline model for the outcome that is additive in the treatment and multivariate spline function of the GPS as the following:

$$f(W_i, \boldsymbol{X}_i) = E[Y_i|W_i, \boldsymbol{X}_i] = \text{logit}^{-1}\Big\{ \boldsymbol{\beta} W_i + h(\boldsymbol{R}(\boldsymbol{X}_i), \boldsymbol{\phi}) \Big\}, \tag{8}$$

where $h(\cdot)$ is a spline function of the GPS indexed by $\boldsymbol{\phi}$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_T]^\top$ are regression coefficients associated with the treatment W_i. The dimension of the spline function $h(\cdot)$ depends on the number of treatments T. Confidence intervals of treatment effect estimates can be obtained using nonparametric bootstrap for RAMS (Hu and Gu, 2021).

In **CIMTx**, RAMS is implemented using the `gam()` function with tensor product smoother `te()` between treatments from the **mgcv** package. Treatment effects can then be estimated by averaging and contrasting the predicted $\hat{f}(w, \boldsymbol{X}_i)$ between treatment groups. The RAMS can be called by setting `method = "RAMS-Multinomial"` and specifying the estimand `estimand = "ATE"` or `estimand = "ATT"`.

```
rams_multi_res <- ce_estimate(y = data$y, x = data$covariates, w = data$w,
                              method = "RAMS-Multinomial", estimand = "ATE",
                              boot = TRUE, nboots = 100, verbose_boot = F)
```

Vector matching Lopez and Gutman (2017) proposed the VM algorithm, which matches individuals with similar vector of the GPS. VM obtains matched sets using a combination of k-means clustering and one-to-one matching with replacement within each cluster strata. Currently, VM is only designed to estimate the ATT effects. In **CIMTx** , VM is implemented via `method = "VM"`. The **CIMTx** does not provide confidence intervals for treatment effect estimates because the authors of this method, Lopez and Gutman (2017), did not provide an approach to estimate the sampling variance of the VM estimator.

To implement VM in **CIMTx**, we set the reference group `reference_trt = 1`, the number of clusters to form using k-means clustering `n_cluster = 3`.

```
vm_res <- ce_estimate(y = data$y, x = data$covariates, w = data$w, method = "VM",
                      estimand = "ATT", reference_trt = 1, n_cluster = 3)
```

The number of matched individuals is also stored in the output list:

```
vm_res$number_matched
```

```
#> 158
```

Targeted maximum likelihood estimation TMLE is a doubly robust approach that combines outcome estimation, IPTW estimation, and a targeting step to optimize the parameter of interest with respect to bias and variance. Rose and Normand (2019) implemented TMLE to estimate the ATE effects of multiple treatments. **CIMTx** calls the R package **tmle** to implement TMLE for the ATE effects. As suggested by Rose and Normand (2019), nonparametric bootstrap is used in **CIMTx** to obtain the confidence interval of the treatment effect estimate.

Calling `method = "TMLE"` implements TMLE in **CIMTx**. We use nonparametric bootstrap to estimate the 95% confidence intervals by setting `boot = TRUE` and `nboots = 100`.

```
tmle_res_boot <- ce_estimate(y = data$y, x = data$covariates, w = data$w, nboots = 100,
                             method = "TMLE", estimand = "ATE", boot = TRUE,
                             sl_library = c("SL.glm", "SL.glmnet", "SL.rpart"))

summary(tmle_res)
```

```
#> $ATE12
#> EST   SE LOWER UPPER
#> RD -0.36 0.04 -0.45 -0.29
#> RR  0.30 0.05  0.21  0.39
#> OR  0.17 0.04  0.11  0.24

#> $ATE13
#> EST   SE LOWER UPPER
#> RD -0.60 0.04 -0.67 -0.51
#> RR  0.20 0.03  0.15  0.28
#> OR  0.06 0.02  0.04  0.10

#> $ATE23
#> EST   SE LOWER UPPER
#> RD -0.24 0.05 -0.34 -0.14
#> RR  0.68 0.06  0.57  0.79
#> OR  0.34 0.09  0.21  0.55
```

Identification of a common support region

Turning to causal identification assumptions. If the positivity assumption (A2) is violated, problems can arise when extrapolating over the areas of the covariate space where common support does not exist. It is important to define a common support region to which the causal conclusions can be generalized. In **CIMTx**, the identification of a common support region is offered in three methods: IPTW, VM and BART.

For IPTW, one strategy is weight truncation, by which extreme weights that fall outside a specified range limit of the weight distribution are set to the range limit. This functionality is offered in **CIMTx** via the `trim_perc` argument. `trim_perc`, which can take two values – one for the lower- and one for the upper-percentile of the weight distribution for trimming. Figure 3 shows the distributions of the weights estimated by the three methods before and after weight trimming at the 5% and 95% of the weight distribution.

```
plot(iptw_multi_res, iptw_sl_res, iptw_gbm_res, iptw_multi_trim_res,
     iptw_sl_trim_res, iptw_gbm_trim_res)
```

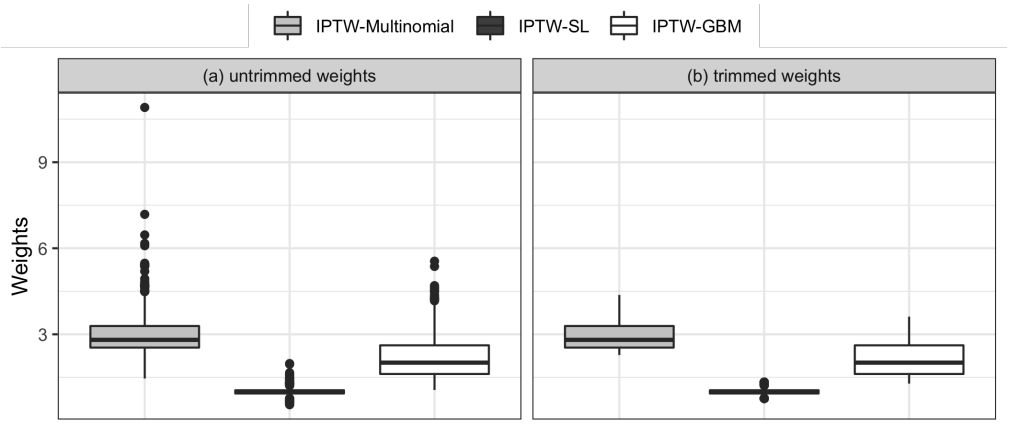


Figure 3: Distributions of the inverse probability of treatment weights estimated by multinomial logistic regression, super learner and generalized boosted models. Panel (a) shows results before weight trimming. Panel (b) displays results after trimming the weights at 5% and 95% of the distribution. Super learner and the generalized boosted models produced less extreme weights compared to multinomial logistic regression.

For VM, Lopez and Gutman (2017) proposed a rectangular support region defined by the maximum value of the smallest GPS and the minimum value of the largest GPS among the treatment groups. Individuals that fall outside the region are discarded from the causal analysis. This feature is automatically implemented with `"VM"` in **CIMTx**.

For BART, Hu et al. (2020a) supplied BART with a strategy to identify a common support region for retaining inferential units, which is to discard individuals with a large variability in their predicted potential outcomes. Specifically, for the ATT effects, any individual i with $W_i = w$ will be discarded if

$$s_i^{f_{w'}} > \max_j\{s_j^{f_w}\}, \forall j : W_j = w, w' \neq w \in \mathscr{W}, \tag{9}$$

where $s_j^{f_w}$ and $s_i^{f_{w'}}$ respectively denote the standard deviation of the posterior distribution of the potential outcomes under treatment $W = w$ and $W = w'$, for a given sample j. For the ATE effects, the discarding rule in equation (9) is applied to each treatment group. Users can implement the discarding rule by setting the `discard` argument in **CIMTx**. Using $ATT_{1|1,2}$ as an example, 5 (`bart_dis_res$n_discard`) individuals in the reference group $w = 1$ were discarded from the simulated data.

```
bart_dis_res <- ce_estimate(y = data$y, x = data$covariates, w = data$w,
                            method = "BART", estimand = "ATT", discard = TRUE,
                            ndpost = 100, reference_trt = 1)
```

Sensitivity analysis for unmeasured confounding

The violation of the ignorability assumption (A3) can lead to biased treatment effect estimates. Sensitivity analysis is useful in gauging how much the causal conclusions will be altered in response to different magnitude of departure from the ignorability assumption. **CIMTx** implements a new flexible sensitivity analysis approach developed by Hu et al. (2022b). This approach first defines a confounding function for any pair of treatments (w, w') as

$$c(w, w', \boldsymbol{x}) = E\left[Y(w) \mid W = w, \boldsymbol{X} = \boldsymbol{x}\right] - E\left[Y(w) \mid W = w', \boldsymbol{X} = \boldsymbol{x}\right]. \tag{10}$$

The confounding function, also viewed as the sensitivity parameter in a sensitivity analysis, directly represents the difference in the mean potential outcomes $Y(w)$ between those treated with $W = w$ and those treated with $W = w'$, who have the same level of $\boldsymbol{x}$. If the ignorability assumption holds, the confounding function will be zero for all $w \in \mathscr{W}$. When treatment assignment is not ignorable, the unmeasured confounding is present and the causal effect estimates using measured $\boldsymbol{X}$ will be biased. Hu et al. (2022b) derived the form of the resultant bias as:

$$\begin{aligned} \text{Bias}(w, w') = &- p_w c(w', w, \boldsymbol{x}) + p_{w'} c(w, w', \boldsymbol{x}) \\ &- \sum_{l: l \in \mathscr{W} \setminus \{w, w'\}} p_l \left\{c(w', l, \boldsymbol{x}) - c(w, l, \boldsymbol{x})\right\}, \end{aligned} \tag{11}$$

where $p_w = P(W = w \mid \boldsymbol{X} = \boldsymbol{x})$, $w \neq w' \in \mathscr{W} = \{1, \ldots, T\}$.

Table 2 demonstrates the plausible assumptions about the confounding functions and their interpretations. There are three ways in which we can specify the prior for the confounding functions: (i) point mass prior; (ii) re-analysis over a range of point mass priors (tipping point); (iii) full prior with uncertainty specified. Since the new sensitivity analysis approach was developed within the Bayesian framework, strategy (iii) offers an advantage of incorporating the statistical uncertainty due to sampling and the uncertainty about the values of the sensitivity parameters. In strategy (i), a fixed value is assumed for the sensitivity parameter. Strategy (ii) expands on strategy (i) and examines how the causal conclusion would change when a range of values are assumed for the sensitivity parameter. We will demonstrate all three cases of prior specifications with `sa()` function in **CIMTx** package. Hu et al. (2022b) further discussed (a) strategies to specify the confounding functions that represent our prior beliefs about the degrees of unmeasured confounding via the remaining variability in the outcomes unexplained by measured $\boldsymbol{X}$ (Hogan et al., 2014); and (b) ways in which the causal effects can be estimated adjusting for the presumed degree of unmeasured confounding.

Table 2: Interpretation of assumed priors on $c(w, w', \boldsymbol{x})$ and $c(w', w, \boldsymbol{x})$ for causal estimands based on the risk difference, assuming the outcome is an adverse event.

Prior assumption		Interpretation and implications of the assumptions
$c(w, w', \boldsymbol{x})$	$c(w', w, \boldsymbol{x})$	
> 0	< 0	Unhealthier individuals are treated with w.
< 0	> 0	Contrary to the above interpretation, unhealthier individuals are treated with w'.
< 0	< 0	The observed treatment allocation between w' and w is beneficial relative to the alternative which reverses treatment assignment for everyone.
> 0	> 0	Contrary to the above interpretation, the observed treatment allocation between w' and w is undesirable relative to the alternative which reverses treatment assignment for everyone.

The proposed sensitivity analysis algorithm proceeds with the following steps (Hu et al., 2022b):

1. Fit a multinomial probit BART model (Kindo et al., 2016) $f^{\text{MBART}}(A \mid X)$ to estimate the GPS, $p_l \equiv P(W = l \mid X = x)\ \forall l \in \mathscr{W}$, for each individual.
2. **for** $w \leftarrow 1$ to T **do**
   Draw M_1 GPS $\tilde{p}_{l1}, \ldots, \tilde{p}_{lM_1}, \forall l \neq w \wedge l \in \mathscr{W}$ from the posterior predictive distribution of $f^{\text{MBART}}(W \mid \boldsymbol{X})$ for each individual.
   for $m \leftarrow 1$ to M_1 **do**
   Draw M_2 values $\eta^*_{lm1}, \ldots, \eta^*_{lmM_2}$ from the prior distribution of each of the confounding functions $c(w, l, \boldsymbol{x})$, for each $l \neq j \wedge l \in \mathscr{W}$.
   end for
   end for

3. Compute the adjusted outcomes, $Y_i^{\text{CF}} \equiv Y_i - \sum_{l \neq w}^{T} P(W_i = l \mid \mathbf{X}_i = \boldsymbol{x}) c(w, l, \boldsymbol{x})$, for each treatment w, for each of $M_1 M_2$ draws of $\{\tilde{p}_{l1}, \eta^*_{l11}, \ldots, \eta^*_{l1M_2}, \ldots, \tilde{p}_{lM_1}, \eta^*_{lM_11}, \ldots, \eta^*_{lM_1M_2}; l \neq w \wedge l \in \mathscr{W}\}$.
4. Fit a BART model to each of $M_1 \times M_2$ sets of observed data with the adjusted outcomes Y^{CF}.
5. Estimate the combined adjusted causal effects and uncertainty intervals by pooling posterior samples across model fits arising from the $M_1 \times M_2$ data sets.

We now demonstrate the Monte Carlo sensitivity analysis approach for unmeasured confounding (Hu et al., 2022b). We first simulate a small dataset in a simple causal inference setting. There are two binary confounders: X_1 is measured and X_2 is unmeasured.

```
set.seed(111)
data_SA <- data_sim(
  sample_size = 100, n_trt = 3,
  x = c("rbinom(1, .5)",  # x1: measured confounder
        "rbinom(1, .4)"), # x2: unmeasured confounder
  lp_y = rep(".2*x1+2.3*x2", 3),# parallel response surfaces
  nlp_y = NULL,
  align = F, # w model is not the same as the y model
  lp_w = c("0.2 * x1 + 2.4 * x2",  # w = 1
           "-0.3 * x1 - 2.8 * x2"),# w = 2
  nlp_w = NULL,
  tau = c(-2, 0, 2),  delta = c(0, 0), psi = 1)
```

Next we implement the sensitivity analysis algorithm step-by-step.

1. Estimate the GPS for each individual. Specifically, we fit a multinomial probit BART model regressing treatment assignment on covariates, using `mbart2()` function from **BART** package. We set the number of posterior draws for the GPS (`m1`) to 50.

   ```
   m1 <- 50; sample_gap <- 10
   w_model <- BART::mbart2(x.train = data_SA$covariates, y.train = data_SA$w,
                           ndpost = m1 * sample_gap)
   ```

2. Then we draw the GPS for each individual from the fitted multinomial probit BART model.

   ```
   gps <- array(w_model$prob.train[seq(1, m1 * sample_gap, sample_gap),],
                dim = c(m1,  # 1st dimension is M1
                        length(unique(data_SA$w)), # 2nd dimension is w
                        dim(data_SA$covariates)[1])) # 3rd dimension is sample size
   dim(gps)

   #> 50 3 100
   ```

   The output of the posterior GPS is a three-dimensional array. The first dimension is the number of posterior draws for the GPS (M_1). The second dimension is the number of treatment W, and the third dimension is the total sample size.

3. Specify the prior distributions and the number of draws (M_2) for the confounding functions $c(w, w', \boldsymbol{x})$. In this illustrative simulation example, we use the true values of the confounding functions within each stratum of x_1. This represents the strategy (i) point mass prior.

   ```
   x1 <- data_SA$covariates[, 1, drop = F]
   x2 <- data_SA$covariates[, 2, drop = F] # x2 as the unmeasured confounder
   w <- data_SA$w
   x1w_data <- cbind(x1, w)
   Y1 <- data_SA$y_true[, 1]
   Y2 <- data_SA$y_true[, 2]
   Y3 <- data_SA$y_true[, 3]
   y <- data_SA$y
   # Calculate the true confounding functions within x1 = 1 stratum
   c_1_x1_1 <- mean(Y1[w == 1 & x1 == 1]) - mean(Y1[w == 2 & x1 == 1]) # c(1,2)
   c_2_x1_1 <- mean(Y2[w == 2 & x1 == 1]) - mean(Y2[w == 1 & x1 == 1]) # c(2,1)
   c_3_x1_1 <- mean(Y2[w == 2 & x1 == 1]) - mean(Y2[w == 3 & x1 == 1]) # c(2,3)
   c_4_x1_1 <- mean(Y1[w == 1 & x1 == 1]) - mean(Y1[w == 3 & x1 == 1]) # c(1,3)
   c_5_x1_1 <- mean(Y3[w == 3 & x1 == 1]) - mean(Y3[w == 1 & x1 == 1]) # c(3,1)
   c_6_x1_1 <- mean(Y3[w == 3 & x1 == 1]) - mean(Y3[w == 2 & x1 == 1]) # c(3,2)
   ```

```
c_x1_1 <- cbind(c_1_x1_1, c_2_x1_1, c_3_x1_1, c_4_x1_1, c_5_x1_1,
                c_6_x1_1)# True confounding functions among x1 = 1
```

The true values of the confounding functions within the stratum $x_1 = 0$ can be calculated in a similar way.

```
c_1_x1_0 <- mean(Y1[w == 1 & x1 == 0]) - mean(Y1[w == 2 & x1 == 0])# c(1,2)
c_2_x1_0 <- mean(Y2[w == 2 & x1 == 0]) - mean(Y2[w == 1 & x1 == 0])# c(2,1)
c_3_x1_0 <- mean(Y2[w == 2 & x1 == 0]) - mean(Y2[w == 3 & x1 == 0])# c(2,3)
c_4_x1_0 <- mean(Y1[w == 1 & x1 == 0]) - mean(Y1[w == 3 & x1 == 0])# c(1,3)
c_5_x1_0 <- mean(Y3[w == 3 & x1 == 0]) - mean(Y3[w == 1 & x1 == 0])# c(3,1)
c_6_x1_0 <- mean(Y3[w == 3 & x1 == 0]) - mean(Y3[w == 2 & x1 == 0])# c(3,2)
c_x1_0 <- cbind(c_1_x1_0, c_2_x1_0, c_3_x1_0, c_4_x1_0, c_5_x1_0, c_6_x1_0)
```

The true values of the confounding functions within the stratum of x_1 can be calculated using the helper function `true_c_fun_cal()` in our package.

```
true_c_fun <- true_c_fun_cal(x = x1, w = w)
```

4. Calculate the confounding function adjusted outcomes with the drawn values of GPS and confounding functions.

```
i <- 1; j <- 1
ycf <- ifelse(
  x1w_data[, "w"] == 1 & x1 == 1,
  # w = 1, x1 = 1
  y - (c_x1_1[i, 1] * gps[j, 2, ] + c_x1_1[i, 4] * gps[j, 3, ]),
  ifelse(
    x1w_data[, "w"] == 1 & x1 == 0,
    # w = 1, x1 = 0
    y - (c_x1_0[i, 1] * gps[j, 2, ] + c_x1_0[i, 4] * gps[j, 3, ]),
    ifelse(
      x1w_data[, "w"] == 2 & x1 == 1,
      # w = 2, x1 = 1
      y - (c_x1_1[i, 2] * gps[j, 1, ] + c_x1_1[i, 3] * gps[j, 3, ]),
      ifelse(
        x1w_data[, "w"] == 2 & x1 == 0,
        # w = 2, x1 = 0
        y - (c_x1_0[i, 2] * gps[j, 1, ] + c_x1_0[i, 3] * gps[j, 3, ]),
        ifelse(
          x1w_data[, "w"] == 3 & x1 == 1,
          # w = 3, x1 = 1
          y - (c_x1_1[i, 5] * gps[j, 1, ] + c_x1_1[i, 6] * gps[j, 2, ]),
          # w = 3, x1 = 0
          y - (c_x1_0[i, 5] * gps[j, 1, ] + c_x1_0[i, 6] * gps[j, 2, ])
        )
      )
    )
  )
)
```

5. Use the adjusted outcomes to estimate the causal effects.

```
bart_mod_sa <- BART::wbart(x.train = x1w_data, y.train = ycf, ndpost = 1000)
predict_1_ate_sa <- BART::pwbart(cbind(x1, w = 1), bart_mod_sa$treedraws)
predict_2_ate_sa <- BART::pwbart(cbind(x1, w = 2), bart_mod_sa$treedraws)
predict_3_ate_sa <- BART::pwbart(cbind(x1, w = 3), bart_mod_sa$treedraws)
RD_ate_12_sa <- rowMeans(predict_1_ate_sa - predict_2_ate_sa)
RD_ate_23_sa <- rowMeans(predict_2_ate_sa - predict_3_ate_sa)
RD_ate_13_sa <- rowMeans(predict_1_ate_sa - predict_3_ate_sa)
predict_1_att_sa <- BART::pwbart(cbind(x1[w == 1,], w = 1), bart_mod_sa$treedraws)
predict_2_att_sa <- BART::pwbart(cbind(x1[w == 1,], w = 2), bart_mod_sa$treedraws)
RD_att_12_sa <- rowMeans(predict_1_att_sa - predict_2_att_sa) # w=1 is the reference
```

Repeat steps 3 and 4 $M_1 \times M_2$ times to form $M_1 \times M_2$ datasets with adjusted outcomes. The uncertainty intervals are estimated by pooling the posteriors across the $M_1 \times M_2$ model fits.

The sa() function implements the sensitivity analysis approach while fitting the $M_1 \times M_2$ models using parallel computation.

```
sa_res <- sa(m1 = 50, m2 = 1, x = x1, y = data_SA$y, w = data_SA$w, ndpost = 100,
             estimand = "ATE", prior_c_function =  true_c_fun, nCores = 3)

summary(sa_res)

#>             EST   SE LOWER UPPER
#> ATE_RD12 -0.44 0.10 -0.64 -0.23
#> ATE_RD13 -0.58 0.11 -0.80 -0.36
#> ATE_RD23 -0.14 0.12 -0.38  0.10
```

We compare the sensitivity analysis results to the naive estimators where we ignore the unmeasured confounder X_2, and to the results where we had access to X_2.

```
bart_with_x2_res <- ce_estimate(y = data_SA$y, x = cbind(x1, x2), w = data_SA$w,
                                method = "BART", estimand = "ATE", ndpost = 100)
bart_without_x2_res <- ce_estimate(y = data_SA$y, x = x1, w = data_SA$w,
                                   method = "BART", estimand = "ATE", ndpost = 100)
```

Figure 4 compares the estimates of $ATE_{1,2}$, $ATE_{2,3}$ and $ATE_{1,3}$ from the three analyses. The sensitivity analysis estimators are similar to the results that could be achieved had the unmeasured confounder X_2 been made available.

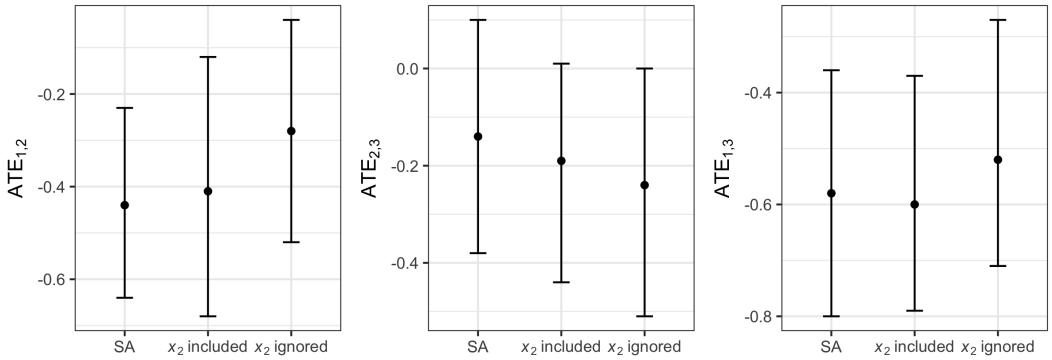


Figure 4: Estimates and 95% credible intervals for $ATE_{1,2}$, $ATE_{2,3}$ and $ATE_{1,3}$

We can also conduct the sensitivity analysis for the ATT effects by setting estimand = "ATT".

```
sa_att_res <- sa(m1 = 50, m2 = 1, x = x1, y = data_SA$y, w = data_SA$w, ndpost = 100,
                 estimand = "ATT", prior_c_function =  true_c_fun, nCores = 1,
                 reference_trt = 1)

summary(sa_att_res)

#>             EST   SE LOWER UPPER
#> ATT_RD12 -0.42 0.10 -0.63 -0.22
#> ATT_RD13 -0.57 0.11 -0.79 -0.35
```

Finally, we demonstrate the sa() function in a more complex data setting with 3 measured confounders and 2 unmeasured confounders.

```
set.seed(1)
data_SA_2 <- data_sim(
  sample_size = 100, n_trt = 3,
  x = c( "rnorm(0, 0.5)",  # x1
         "rbeta(2, .4)",   # x2
         "runif(0, 0.5)",  # x3
         "rweibull(1, 2)", # x4 as one of the unmeasured confounders
         "rbinom(1, .4)" ),  # x5 as one of the unmeasured confounders
  lp_y = rep(".2*x1 + .3*x2 - .1*x3 - 1.1*x4 - 1.2*x5", 3),
  nlp_y  = rep(".7*x1*x1  - .1*x2*x3", 3), # parallel response surfaces
  align = FALSE,
```

```
lp_w =  c(".4*x1 + .1*x2 - 1.1*x4 + 1.1*x5",  # w = 1
          ".2*x1 + .2*x2 - 1.2*x4 - 1.3*x5"), # w = 2,
nlp_w = c("-.5*x1*x4 - .1*x2*x5",  # w = 1
          "-.3*x1*x4 + .2*x2*x5"), # w = 2,
tau = c(0.5,-0.5,0.5), delta = c(0.5,0.5), psi = 2)
```

We have demonstrated the strategy (i) point mass prior, and now show how strategy (ii) re-analysis over a range of point mass priors and (iii) full prior with uncertainty specified can be used. For strategy (ii), we can specify the grid of the specific confounding function using `seq()` function. In the following example, we will set $c(1,3)$ as a grid of 5 negative numbers from -0.6 to 0 with an increment of 0.15, and set $c(3,1)$ as a grid of 5 positive numbers from 0 to 0.6 with an increment of 0.15. This specification encodes our belief that unhealthier (suppose the outcome is death) individuals are treated with treatment option 3 (see Table 2) because those receiving $w = 1$ would have lower probability of death to either treatment. The other confounding functions are drawn from a uniform distribution (strategy (iii)).

```
c_grid <- c("runif(-.6, 0)", "runif(0,.6)", "runif(-.6,0)", # c(1,2), c(2,1), c(2,3)
            "seq(-.6, 0,.15)", "seq(0,.6,.15)", "runif(0,.6)") # c(1,3), c(3,1), c(3,2)
SA_grid_res <- sa(y = data_SA_2$y, w = data_SA_2$w, x = data_SA_2$covariates[,-c(4,5)],
                  prior_c_function = c_grid, m1 = 1, nCores = 3, estimand = "ATE")
```

The sensitivity analysis results can be visualized via a contour plot. Figure 5 shows how the estimate of $ATE_{1,3}$ would change under different values of the two confounding functions $c(1,3,\boldsymbol{x})$ and $c(3,1,\boldsymbol{x})$. Under the assumption that unhealthier patients are treated with $w = 3$, when the effect of unmeasured confounding increases (moving up along the $-45°$ line), the beneficial treatment effect associated with $w = 3$ becomes more pronounced evidenced by larger estimates of $ATE_{1,3}$.

```
plot(SA_grid_res, ATE = "1,3")
```

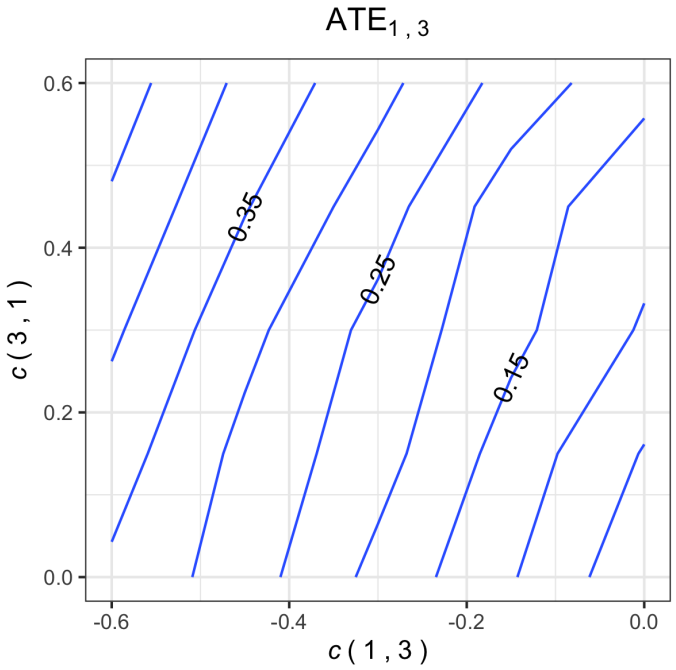


Figure 5: Contour plot of the confounding function adjusted $ATE_{1,3}$. The blue lines report the adjusted causal effect estimates corresponding to pairs of values for $c(1,3)$ and $c(3,1)$ spaced on a grid $(-0.6,0) \times (0,0.6)$ incremented by 0.15, and under the prior distributions: $c(1,2), c(2,3) \sim U(-0.6,0); c(2,1), c(2,3) \sim U(0,0.6)$.

4 Discussion

We contribute a comprehensive R package **CIMTx** suitable for causal analysis of observational data with multiple treatments and a binary outcome. In this package, we introduce six methods for the

estimation of causal effects, including both the classical approaches and machine learning based methods. Drawing causal inference from non-experimental data inevitably involves structural causal assumptions. **CIMTx** offers a unique set of features to address two key assumptions: positivity and ignorability, using appropriate estimation procedures. Additionally, the **CIMTx** package provides guidance to readers on how to simulate data possessing the data characteristics in the multiple treatment setting. Detailed step-by-step examples are provided to demonstrate all methods. The current version of the **CIMTx** package focuses on binary outcomes. For future research, developing methods and R packages for causal inferences with more complex outcomes such as censored survival outcomes (Hu et al., 2022a) could be a worthwhile contribution.

Bibliography

P. C. Austin. Variance estimation when using inverse probability of treatment weighting (IPTW) with survival analysis. *Statistics in Medicine*, 35(30):5642–5655, 2016. [p219]

H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010. [p220]

S. R. Cole and M. A. Hernán. Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, 168(6):656–664, 2008. [p219]

M. Erik von Elm, D. G. Altman, M. Egger, et al. The Strengthening the Reporting of Observational Studies in Epidemiology (STROBE) statement: guidelines for reporting observational studies. *Annals of Internal Medicine*, 147(8):573—-577, 2007. [p213]

P. Feng, X.-H. Zhou, Q.-M. Zou, M.-Y. Fan, and X.-S. Li. Generalized propensity score for estimating the average treatment effect of multiple treatments. *Statistics in Medicine*, 31(7):681–697, 2012. [p213, 219]

C. Fong, M. Ratkovic, and K. Imai. *CBPS: Covariate Balancing Propensity Score*, 2021. URL https://CRAN.R-project.org/package=CBPS. R package version 0.22. [p214]

J. M. Franklin, W. Eddings, P. C. Austin, E. A. Stuart, and S. Schneeweiss. Comparing the performance of propensity score methods in healthcare database studies with rare outcomes. *Statistics in Medicine*, 36(12):1946–1963, 2017. [p220]

N. Greifer. *optweight: Targeted Stable Balancing Weights Using Optimization*, 2019. URL https://CRAN.R-project.org/package=optweight. R package version 0.2.5. [p214]

N. Greifer. *WeightIt: Weighting for Covariate Balance in Observational Studies*, 2020. URL https://CRAN.R-project.org/package=WeightIt. R package version 0.10.2. [p214]

J. L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011. [p213, 220]

J. W. Hogan, M. J. Daniels, and L. Hu. A bayesian perspective on assessing sensitivity to assumptions about unobserved data. In G. Molenberghs, G. Fitzmaurice, M. G. Kenward, A. Tsiatis, and G. Verbeke, editors, *Handbook of Missing Data Methodology*, chapter 18, pages 405–434. Boca Raton, FL: CRC Press, 2014. [p223]

P. W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396): 945–960, 1986. [p217]

D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952. [p218]

L. Hu and C. Gu. Estimation of causal effects of multiple treatments in healthcare database studies with rare outcomes. *Health Services and Outcomes Research Methodology*, 21(3):287–308, 2021. [p213, 220]

L. Hu, C. Gu, M. Lopez, J. Ji, and J. Wisnivesky. Estimation of causal effects of multiple treatments in observational studies with a binary outcome. *Statistical Methods in Medical Research*, 29(11): 3218–3234, 2020a. [p213, 217, 220, 222]

L. Hu, B. Liu, J. Ji, and Y. Li. Tree-based machine learning to identify and understand major determinants for stroke at the neighborhood level. *Journal of the American Heart Association*, 9(22):e016745, 2020b. [p220]

L. Hu, J. Ji, and F. Li. Estimating heterogeneous survival treatment effect in observational data using machine learning. *Statistics in Medicine*, 40(21):4691–4713, 2021a. [p215, 220]

L. Hu, J.-Y. Joyce Lin, and J. Ji. Variable selection with missing data in both covariates and outcomes: Imputation and machine learning. *Statistical Methods in Medical Research*, 30(12):2651–2671, 2021b. [p213]

L. Hu, J.-Y. J. Lin, K. Sigel, and M. Kale. Estimating heterogeneous survival treatment effects of lung cancer screening approaches: A causal machine learning analysis. *Annals of Epidemiology*, 62:36–42, 2021c. [p220]

L. Hu, J. Ji, R. D. Ennis, and J. W. Hogan. A flexible approach for causal inference with multiple treatments and clustered survival outcomes. *arXiv preprint*, 2022a. arXiv:2202.08318. [p228]

L. Hu, J. Zou, C. Gu, J. Ji, M. Lopez, and M. Kale. A flexible sensitivity analysis approach for unmeasured confounding with multiple treatments and a binary outcome with application to SEER-Medicare lung cancer data. *The Annals of Applied Statistics*, 16(2):1014–1037, 2022b. [p213, 223, 224]

G. W. Imbens. The role of the propensity score in estimating dose-response functions. *Biometrika*, 87(3): 706–710, 2000. [p219]

G. W. Imbens and D. B. Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015. [p217]

J. D. Kang, J. L. Schafer, et al. Demystifying double robustness: A comparison of alternative strategies for estimating a population mean from incomplete data. *Statistical Science*, 22(4):523–539, 2007. [p219]

B. P. Kindo, H. Wang, and E. A. Peña. Multinomial probit bayesian additive regression trees. *Stat*, 5(1): 119–131, 2016. [p223]

J. Kruschke. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014. [p218]

B. K. Lee, J. Lessler, and E. A. Stuart. Weight trimming and propensity score weighting. *PLOS One*, 6 (3):e18174, 2011. [p219]

A. Linden, S. D. Uysal, A. Ryan, and J. L. Adams. Estimating causal effects for multivalued treatments: a comparison of approaches. *Statistics in Medicine*, 35(4):534–552, 2016. [p213, 217]

R. J. Little. Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics*, 6(3): 287–296, 1988. [p219]

M. J. Lopez and R. Gutman. Estimation of causal effects with multiple treatments: a review and new ideas. *Statistical Science*, 32(3):432–454, 2017. [p213, 221, 222]

D. F. McCaffrey, B. A. Griffin, D. Almirall, M. E. Slaughter, R. Ramchand, and L. F. Burgette. A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in Medicine*, 32(19):3388–3414, 2013. [p213, 219]

G. Ridgeway, D. McCaffrey, A. Morral, B. A. Griffin, L. Burgette, and M. Cefalu. *twang: Toolkit for Weighting and Analysis of Nonequivalent Groups*, 2020. URL `https://CRAN.R-project.org/package=twang`. R package version 1.6. [p214]

S. Rose and S.-L. Normand. Double robust estimation for multiple unordered treatments and clustered observations: Evaluating drug-eluting coronary artery stents. *Biometrics*, 75(1):289–296, 2019. [p213, 221]

D. B. Rubin. The use of matched sampling and regression adjustment to remove bias in observational studies. *Biometrics*, pages 185–203, 1973. [p213, 217]

D. B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974. [p217]

D. B. Rubin. Randomization analysis of experimental data: The Fisher randomization test comment. *Journal of the American Statistical Association*, 75(371):591–593, 1980. [p217]

M. J. Van der Laan, E. C. Polley, and A. E. Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007. [p219]

T. Zhou, G. Tong, F. Li, L. Thomas, and F. Li. *PSweight: Propensity Score Weighting for Causal Inference with Observational Studies and Randomized Trials*, 2020. URL https://CRAN.R-project.org/package=PSweight. R package version 1.1.2. [p214]

Liangyuan Hu
Rutgers University School of Public Health
Department of Biostatistics and Epidemiology
683 Hoes Lane West
Piscataway, NJ 08854, United States of America
liangyuan.hu@rutgers.edu

Jiayi Ji
Rutgers University School of Public Health
Department of Biostatistics and Epidemiology
683 Hoes Lane West
Piscataway, NJ 08854, United States of America
jj869@sph.rutgers.edu

logitFD: an R package for functional principal component logit regression

by Manuel Escabias, Ana M. Aguilera and Christian Acal

Abstract The functional logit regression model was proposed by Escabias et al. (2004) with the objective of modeling a scalar binary response variable from a functional predictor. The model estimation proposed in that case was performed in a subspace of $L^2(T)$ of squared integrable functions of finite dimension, generated by a finite set of basis functions. For that estimation it was assumed that the curves of the functional predictor and the functional parameter of the model belong to the same finite subspace. The estimation so obtained was affected by high multicollinearity problems and the solution given to these problems was based on different functional principal component analysis. The **logitFD** package introduced here provides a toolbox for the fit of these models by implementing the different proposed solutions and by generalizing the model proposed in 2004 to the case of several functional and non-functional predictors. The performance of the functions is illustrated by using data sets of functional data included in the **fda.usc** package from R-CRAN.

1 Introduction

A functional variable is that whose values depend on a continuous magnitude such as time. They are functional in the sense that they can be evaluated at any time point of the domain, instead of the discrete way, in which they were originally measured or observed (see for example Ramsay and Silverman, 2005). Different approaches have been used for the study of functional data, among others, the nonparametric methods proposed by Müller and Stadtmüller (2005) and Ferraty and Vieu (2006) or the basis expansion methods considered in Ramsay and Silverman (2005). Most multivariate statistical techniques have been extended for functional data, whose basic theory and inferential aspects are collected in recent books by Horvath and Kokoszka (2012), Zhang (2014) and Kokoszka and Reimherr (2018). The basic tools to reduce the dimension of the functional space to which the curves belong, are Functional Principal and Independent Component Analysis (FPCA) (Ramsay and Silverman, 2005; Acal et al., 2020; Vidal et al., 2021) and Functional Partial Least Squares (FPLS) (Preda and Saporta, 2005; Aguilera et al., 2010; Aguilera et al., 2016).

In the last decade of the XXth century and the first decade of XXIth century, where functional data methods began to be developed, there was no adequate software available for using and fitting functional data methods. In fact, nowadays classical statistical software like SPSS, STATA,... do not have a toolbox for functional data analysis. The development of object-oriented software like R, Matlab or S-plus and the great activity of scientific community in this field has made possible to emerge different packages mainly in R for using functional data analysis (FDA) methods. Every package is designed from the point of view followed by its developer and the method used to fit functional data methods. For example Febrero-Bande and Oviedo (2012) used nonparametric methods in their **fda.usc** package, Ramsay et al. (2009) designed their **fda** package under basis expansion philosophy, Principal Analysis by Conditional Estimation (PACE) algorithm (see Zhu et al., 2014) was used for curves alignment, PCA and regression in the **fdasrvf** package (see `https://cran.r-project.org/web/packages/fdasrvf/index.html`). Recently Fabian Scheipl has summarized the available R packages for FDA (see `https://cran.r-project.org/web/views/FunctionalData.html`).

This paper is devoted to **logitFD** an R package for fitting the different functional principal component logit regression approaches proposed by Escabias et al. (2004). Functional logit regression is a functional method for modeling a scalar binary response variable in different situations: firstly, from one single functional variable as predictor; secondly, from several functional variables as predictors; and thirdly, from several functional and nonfunctional variables as predictors which is the most general case. There exist some R functions with this objective as the `fregre.glm` function of **fda.usc** package (see `https://rpubs.com/moviedo/fda_usc_regression`). Different to the former the methods proposed by Escabias et al. (2004), and developed in **logitFD**, are basis expansion based methods what makes the logit model suffer from multicollinearity. The proposed solutions were based on different functional principal components analysis: ordinary FPCA and filtered FPCA (see Escabias et al., 2014). These models have been successfully applied to solve environmental problems (Aguilera et al., 2008b; Escabias et al., 2005; Escabias et al., 2013) and classification problems in food industry (Aguilera-Morillo and Aguilera, 2015). Extensions for the case of sparse and correlated data or generalized models have been also studied (James, 2002; Müller and Stadtmüller, 2005; Aguilera-Morillo et al., 2013; Mousavi and Sørensen, 2018; Tapia et al., 2019; Bianco et al., 2021).

This package adopts **fda**'s package philosophy of basis expansion methods of Ramsay et al. (2009)

and it is designed to use objects inherited from the ones defined in **fda** package. For this reason **fda** package is required for **logitFD**. The package consists of four functions that fit a functional principal component logit regression model in four different situations

- Filtered functional principal components of functional predictors, included in the model according to their variability explained power.
- Filtered functional principal components of functional predictors, included in the model automatically according to their prediction ability by stepwise methods.
- Ordinary functional principal components of functional predictors, included in the model according to their variability explained power.
- Ordinary functional principal components of functional predictors, included in the model automatically according to their prediction ability by stepwise methods.

The designed functions of our package use as input the `fd` objects given by **fda** package and also provide as output `fd` objects among others elements.

This paper is structured as follows: after this introduction, the second section shows the generalities of the package with the needed definitions and objects of functional data analysis, functional logit regression and extended functional logit regression, third and fourth sections board ordinary and filtered functional principal component logit regression, respectively. In fifth section ordinary and filtered functional principal components logit regression is addressed by including functional principal components according prediction ability by stepwise methods. In every section a summary of the theoretical aspects of the involved models is shown with a practical application with functional data contained in **fda.usc** package (Febrero-Bande and Oviedo, 2012).

2 logitFD package: general statements

Functional data analysis

A functional data set is a set of curves $\{x_1(t), \ldots, x_n(t)\}$, with t in a real interval T ($t \in T$). Each curve can be observed at different time points of its argument t as $x_i = (x_i(t_0), \ldots, x_i(t_{m_i}))'$ for the set of times $t_0, \ldots, t_{m_i}$, $i = 1, \ldots, n$ and these are not necessarily the same for each curve.

Basis expansion methods assume that the curves belong to a finite dimensional space generated by a basis of functions $\{\phi_1(t), \ldots, \phi_p(t)\}$ and so they can be expressed as

$$x_i(t) = \sum_{j=1}^{p} a_{ij}\phi_j(t), \; i = 1, \ldots, n. \tag{1}$$

The functional form of the curves is determined when the basis coefficients $a_i = \left(a_{i1}, \ldots, a_{ip}\right)'$ are known. These can be obtained from the discrete observations either by least squares or by interpolation methods (see, for example, Escabias et al., 2005 and Escabias et al., 2007).

Depending on the characteristics of the curves and the observations, various types of basis can be used (see, for example, Ramsay and Silverman, 2005). In practice, those most commonly used are, on the one had, the basis of trigonometric functions for regular, periodic, continuous and differentiable curves, and on the other hand, the basis of B-spline functions, which provides a better local behavior (see De Boor, 2001). In **fda** package the type of basis used are B-spline basis, constant basis, exponential basis, Fourier basis, monomial basis, polygonal basis and power basis (Ramsay et al., 2009). Due to **logitFD** package use `fd` objects from **fda** package, the same types of basis can be used.

In order to illustrate the use of **logitFD** package we are going to use `aemet` data included in **fda.usc** package of Febrero-Bande and Oviedo (2012). As can be read in the package manual, `aemet` data consist of meteorological data of 73 Spanish weather stations. This data set contains functional and nonfunctional variables observed in all the 73 weather stations. The information we are going to use to illustrate the use of our **logitFD** package is the following:

- `aemet$temp`: matrix with 73 rows and 365 columns with the average daily temperature for the period 1980-2009 in Celsius degrees for each weather station.
- `aemet$logprec`: matrix with 73 rows and 365 columns with the average logarithm of precipitation for the period 1980-2009 for each weather station. We are going to use the proper precipitation, that is, `exp(aemet$logprec)`
- `aemet$wind.speed`: matrix with 73 rows and 365 columns with the average wind speed for the period 1980-2009 for each weather station.

- `aemet$df[,c("ind","altitude","longitude","latitude")]`: data frame with 73 rows and 4 columns with the identifications code of each weather station, the altitude in meters over sea level and longitude and latitude of each weather station.

The problem with daily data is that they are too wiggly so if we need smooth curves with few basis functions, the loose of information is big. So, in order to illustrate the use of **logitFD** package we are going to use mean monthly data. So for each one of the previously defined matrices we consider mean monthly data. On the other hand, `logprec` is also a very wiggly data set, so we considered their exponential. So the final data sets considered were the following:

- `TempMonth`: matrix with 73 rows and 12 columns with the mean monthly temperature of `aemet$temp`.
- `PrecMonth`: matrix with 73 rows and 12 columns with the mean monthly exponential of `aemet$logprec`.
- `WindMonth`: matrix with 73 rows and 12 columns with the mean monthly wind speed of `aemet$wind.speed`.

We are going to consider as binary response variable that variable with values: 1 if a weather station is located in the north of Spain (above Madrid, the capital of Spain, and located in the geographic center of the country) and 0 otherwise (stations of the south). Our objective will be to model the location of weather stations (north/south) from their meteorological information. This is a really artificial problem trying to explain the climate characteristics of Spanish weather stations classified according to their geographical location. Let us observe that only latitude is enough to determine the location of a weather station in the sense we are defining. In fact, latitude allows complete separation that makes the estimation of the logit model impossible (see for example Hosmer et al., 2013).

The steps for reading data would be

```
library(fda.usc)
data(aemet)
Temp <- aemet$temp$data
Prec <- exp(aemet$logprec$data)
Wind <- aemet$wind.speed$data
StationsVars <- aemet$df[,c("ind","altitude","longitude","latitude")]
StationsVars$North <- c(1,1,1,1,0,0,0,0,1,1,0,0,0,0,0,1,1,1,0,0,1,0,0,0,0,1,0,0,1,1,1,1,1,
0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1)
```

and the transformations to consider mean monthly data from daily data only for Temperature

```
 TempMonth <- matrix(0,73,12)
for (i in 1:nrow(TempMonth)){
  TempMonth[i,1] <- mean(Temp[i,1:31])
  TempMonth[i,2] <- mean(Temp[i,32:59])
  TempMonth[i,3] <- mean(Temp[i,60:90])
  TempMonth[i,4] <- mean(Temp[i,91:120])
  TempMonth[i,5] <- mean(Temp[i,121:151])
  TempMonth[i,6] <- mean(Temp[i,152:181])
  TempMonth[i,7] <- mean(Temp[i,182:212])
  TempMonth[i,8] <- mean(Temp[i,213:243])
  TempMonth[i,9] <- mean(Temp[i,244:273])
  TempMonth[i,10] <- mean(Temp[i,274:304])
  TempMonth[i,11] <- mean(Temp[i,305:334])
  TempMonth[i,12] <- mean(Temp[i,335:365])
  }
```

The rest of matrices (`PrecMonth` and `WindMonth`) were obtained in the same way.

logitFD is an R package for fitting functional principal component logit regression based on ordinary and filtered functional principal components described in previous sections. As was stated in the introduction, this package uses **fda**'s package philosophy of basis expansion methods and it is designed to use objects inherited from the ones defined in **fda** package. For this reason **fda** package is required for **logitFD**. The R functions designed in our package use as input the `fd` objects given by **fda** package and also provide as output `fd` objects among others elements. In order to use our package it is assumed that the reader manage with **fda** package, its objects and functions.

Let us begin with a brief explanation of the **fda** objects required in our proposal. **fda** package builds, from discrete observations of curves, an `fd` object (named `fdobj`) that will be used by **logitFD** for its

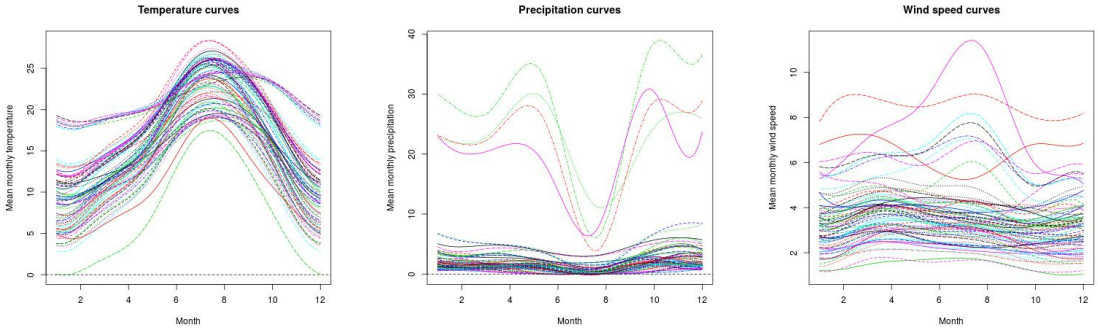


Figure 1: Curves of mean monthly Temperature (left), Precipitation (middle) and Wind Speed (right) registered in Spanish weather stations along one year. Data set `aemet` from **fda.usc** package. Numbers 1, 2, . . . , 12 in the horizontal axis refer to months January, February, . . . , December respectively.

methods. So, let $X_{n\times m} = (x_i(t_k)),\ i = 1,\ldots,n;\ k = 1,\ldots,m$ be the matrix of discrete observations of curves $x_1(t), x_2(t), \ldots, x_n(t)$ at the same time points $t_1, t_2, \ldots, t_m$. An `fd` object is an `R` list with elements:

- `coefs`: the matrix of basis coefficients.
- `basis`: an object of type `basis` with the information needed to build the functional form of curves based on basis expansion methods explained before. Depending on the selected basis the list of objects that contains the `basis` object can be different (see **fda** reference manual).
- `fdnames`: a list containing names for the arguments, function values and variables. This argument is not necessary.

The matrix of basis coefficients $A_{n\times p} = (a_{ij}),\ i = 1,\ldots,n;\ j = 1,\ldots,p$ (`coefs`) of all curves are obtained by least squares as $A^T = \left(\Phi^T\Phi\right)^{-1}\Phi^T X^T$ where $\Phi_{m\times p} = (\phi_j(t_k)),\ j = 1,\ldots,p;\ k = 1,\ldots,m$ is the matrix of basis functions evaluated at sampling points.

The `basis` object allows the basis expansion (1) of curves. We consider for aemet data these two basis:

- 7-length Fourier basis for Temperature.
- 8-length cubic B-spline basis for Precipitation and Wind

The `R` parameters needed to define the basis object depend on the type of basis used (see fda R reference manual). Fourier basis only needs the interval where basis functions are defined and the dimension of the basis. B-spline basis needs also the degree of polynomials that define the basis functions. The default degree is 3.

The code to create the used basis have been

```
FourierBasis <- create.fourier.basis(rangeval = c(1,12),nbasis=7)
BsplineBasis <- create.bspline.basis(rangeval = c(1,12),nbasis=8)
```

The main function of **fda** package that provides the `fdobj` object from discrete data in a matrix is `Data2fd` function (see **fda** reference manual). Our `fdobj` were obtained with the code

```
TempMonth.fd <- Data2fd(argvals = c(1:12), y=t(TempMonth),basisobj = FourierBasis)
PrecMonth.fd <- Data2fd(argvals = c(1:12), y=t(PrecMonth),basisobj = BsplineBasis)
WindMonth.fd <- Data2fd(argvals = c(1:12), y=t(WindMonth),basisobj = BsplineBasis)
```

An `fdobj` allows plotting all curves by using the `R plot()` command. The functional data so obtained can be seen in Figure 1.

Functional logit regression model

In order to understand how the functions of the **logitFD** work, let summarize the theoretical aspects of the models involved.

Let Y be a binary response random variable and let $\{X_1(t), X_2(t), \ldots, X_R(t) : t \in T\}$ be a set of functional covariates related to Y. Let $x_{11}(t), \ldots, x_{n1}(t), \ldots, x_{1R}(t), \ldots, x_{nR}(t)$ be R samples of curves of the functional predictors that can be summarized by columns in a matrix of curves

$$\begin{pmatrix} x_{11}(t) & x_{12}(t) & \cdots & x_{1R}(t) \\ x_{21}(t) & x_{22}(t) & \cdots & x_{2R}(t) \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1}(t) & x_{n2}(t) & \cdots & x_{nR}(t). \end{pmatrix} \tag{2}$$

Let $y_1, \ldots, y_n$ be a sample of the binary response associated with the curves ($y_i \in \{0, 1\}$), then the functional logit model in terms of the functional predictors is formulated as

$$y_i = \pi_i + \varepsilon_i = \pi(x_{i1}(t), \ldots, x_{iR}(t)) + \varepsilon_i \Leftrightarrow \pi_i = \frac{\exp\{l_i\}}{1 + \exp\{l_i\}},\ i = 1, \ldots, n, \tag{3}$$

where $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)'$ is the vector of independent centered random errors, with unequal variances and Bernoulli distribution, and l_i (known as logit transformations) are modelized from functional predictors as

$$l_i = \ln\left[\frac{\pi_i}{1-\pi_i}\right] = \alpha + \int_T x_{i1}(t)\beta_1(t)\,dt + \int_T x_{i2}(t)\beta_2(t)\,dt + \cdots + \int_T x_{iR}(t)\beta_R(t)\,dt. \tag{4}$$

This model has R functional parameters to be estimated $\beta_1(t), \ldots, \beta_R(t)$. If we consider that the curves of each functional predictor belong to a finite space generated by a basis of functions as in (1) and that the corresponding functional parameter belongs to the same space (same basis for each pair $(X_r(t), \beta_r(t))$, $r = 1, \ldots, R$)

$$\beta_r(t) = \sum_{l=1}^{p_r} \beta_{rl}\phi_{rl}(t),\ r = 1, \ldots, R, \tag{5}$$

the functional logit model in terms of the logit transformations is expressed in matrix form as

$$L = \mathbf{1}\alpha + A_1\Psi_1\beta_1 + A_2\Psi_2\beta_2 + \cdots + A_R\Psi_R\beta_R, \tag{6}$$

where

- $L = (l_1, \ldots, l_n)'$ is the vector of logit transformations.
- $(\mathbf{1} \mid A_1\Psi_1 \mid A_2\Psi_2 \mid \cdots \mid A_R\Psi_R)$ is the design matrix, and $\mid$ indicating the separation between the boxes of the matrix.
- $\mathbf{1} = (1, \ldots, 1)'$ is a $n-$length vector of ones.
- $\Psi_r = (\psi_{jkr})$, $r = 1, \ldots, R$ is the matrix whose entries are the inner products between basis functions of the space where curves belong to

$$\psi_{jkr} = <\phi_{jr}(t), \phi_{kr}(t)> = \int_T \phi_{jr}(t)\phi_{kr}(t)dt,\ j, k = 1, \ldots, p_r;\ r = 1, \ldots, R. \tag{7}$$

- A_r, $r = 1, \ldots, R$ is the matrix of basis coefficients as rows of sample curves of the space where curves belong to.
- $\beta_r = (\beta_{r1}, \ldots, \beta_{rp_r})'$, $r = 1, \ldots, R$ are the basis coefficients of the functional parameter $\beta_r(t)$, $r = 1, \ldots, R$.

Let us observe that each functional predictor (and functional parameter) can be expressed in terms of a different type of basis and different number of basis functions.

This functional logit model provides severe multicollinearity problems as was stated in Escabias et al. (2004) for the case of a single functional predictor that was the original formulation of the model.

Extended functional logit model: several functional and nonfunctional predictors

We can finally formulate the functional logit model in terms of more than one functional predictor and non-functional ones. So let Y be a binary response variable and let $\{X_1(t), X_2(t), \ldots, X_R(t) : t \in T\}$ be a set of functional covariates related to Y and $U_1, U_2, \ldots, U_S$ a set of non-functional predictors. Let

us consider the sample of curves (2) and

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1S} \\ u_{21} & u_{22} & \cdots & u_{2S} \\ \cdots & \cdots & \cdots & \cdots \\ u_{n1} & u_{n2} & \cdots & u_{nS} \end{pmatrix},$$

the sample of observations of nonfunctional predictors. Let $y_1, \ldots, y_n$ be a sample of the response associated with the curves. Then the model is expressed in terms of logit transformations as

$$l_i = \alpha + \int_T x_{i1}(t)\,\beta_1(t)\,dt + \cdots + \int_T x_{iR}(t)\,\beta_R(t)\,dt + u_{i1}\delta_1 + \cdots + u_{iS}\delta_S,\ i = 1, \ldots, n. \tag{8}$$

Now the model has R functional parameters to estimate $\beta_1(t), \ldots, \beta_R(t)$ and S nonfunctional parameters $\delta_1, \ldots, \delta_S$. As in the previous case, each functional predictor and functional parameter can be expressed in terms of a different type of basis and different number of basis functions as in (1) and (5). We consider again the same basis for each pair $(X_r(t), \beta_r(t))$, $r = 1, \ldots, R$. The functional logit model in terms of the logit transformations is expressed in matrix form as

$$L = \mathbf{1}\alpha + A_1\Psi_1\beta_1 + A_2\Psi_2\beta_2 + \cdots + A_R\Psi_R\beta_R + U_1\delta_1 + \cdots + U_S\delta_S.$$

This model has as only difference with respect the previous one the design matrix of the model $(\mathbf{1} \mid A_1\Psi_1 \mid A_2\Psi_2 \mid \cdots \mid A_R\Psi_R \mid U_1 \mid \cdots \mid U_S)$, where $U_1, \ldots, U_S$ represent the columns of observations of the nonfunctional predictors, and a set of scalar parameters $\delta_1, \ldots, \delta_S$. As in the previous case, this model has multicollinearity problems.

3 Ordinary functional principal components logit regression

The proposed solution to solve the multicollinearity problems in Escabias et al. (2004) for the single model (only one functional predictor) was to use as predictors a set of functional principal components. Let us briefly remember the functional principal component analysis principles.

Let $x_1(t), \ldots, x_n(t)$ be a set of curves with mean curve and covariance surface respectively

$$\overline{x}(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(t),\ C(s,t) = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i(s) - \overline{x}(s)\right)\left(x_i(t) - \overline{x}(t)\right).$$

Functional principal components are defined as

$$\xi_{ij} = \int_T \left(x_i(t) - \overline{x}(t)\right) f_j(t)\,dt,\ f_j(t) = \sum_{k=1}^{p} F_{jk}\phi_k(t),\ j = 1, \ldots, p;\ i = 1, \ldots, n.$$

In this formulation it is assumed that curves are expressed as in (1), and, as a consequence, the eigenfunctions $f_j(t)$, that define the functional principal components, are also basis expansion expressed, being the basis coefficients F_j the eigenvectors of $A\Psi^{1/2}$ matrix (see Ocaña et al., 2007). For a more general and detailed situation see Ramsay and Silverman (2005). The original curves can be expressed in terms of the functional principal components as

$$x_i(t) = \overline{x}(t) + \sum_{j=1}^{p} \xi_{ij} f_j(t) = \overline{x}(t) + \sum_{j=1}^{p}\sum_{k=1}^{p} \xi_{ij} F_{jk}\phi_k(t),\ i = 1, \ldots, n.$$

If a reduced set of functional principal components is considered, the original curves can be approximated by

$$x_i(t) \simeq \overline{x}(t) + \sum_{j=1}^{q<p} \xi_{ij} f_j(t) = \overline{x}(t) + \sum_{j=1}^{q<p}\sum_{k=1}^{p} \xi_{ij} F_{jk}\phi_k(t),\ i = 1, \ldots, n \tag{9}$$

The quality of this approximation will depend on the percentage of explained variability that acumulates the first q functional principal components, given by

$$\frac{\sum_{j=1}^{q}\lambda_j}{\sum_{j=1}^{p}\lambda_j}.$$

The ordinary functional principal components logit regression solution to solve the multicollinearity problems of the functional logistic regression model consists of considering a functional principal component expansion of each sample curve for each functional predictor and turning the functional

model into a multivariate one whose covariates are the considered functional principal components. The number of principal components required can be different in each functional predictor, but the same for all curves of a specific functional predictor.

In order to get an estimation of the functional parameter for the case of a single functional covariate, by considering the principal component expansion of curves, the logit model adopts the following expression

$$\begin{aligned} l_i &= \alpha + \int_T \left(\overline{x}(t) + \sum_{j=1}^{p} \xi_{ij} f_j(t) \right) \beta(t)\, dt = \alpha + \int_T \overline{x}(t)\beta(t)\, dt + \sum_{j=1}^{p} \xi_{ij} \int_T f_j(t)\, \beta(t)\, dt, \\ &= \gamma_0 + \sum_{j=1}^{p} \xi_{ij}\gamma_j,\ i = 1, \ldots, n. \end{aligned}$$

These expressions enables to express the basis coefficients of the functional parameter and the intercept parameter of the logit model in terms of the parameters estimated from the functional principal components of the curves.

$$\alpha = \gamma_0 - \int_T \overline{x}(t)\beta(t)\, dt = \alpha + (\overline{a}_1, \ldots, \overline{a}_p)\Psi(\beta_1, \ldots \beta_p)', \tag{10}$$

$$(\beta_1, \ldots, \beta_p)' = \Psi F(\gamma_1, \ldots, \gamma_p)' \tag{11}$$

with $\Psi = \left(\psi_{jk}\right)$ being the inner products between the basis functions (as in (7)) and F the orthogonal matrix of basis coefficients of principal component curves shown in (9).

If we consider the principal component expansion of curves in terms of a reduced set of functional principal components we can get an estimation of the basis coefficients of the functional parameter whose accuracy depends on the accumulated variability of the selected principal components (see Escabias et al., 2004).

So, if we denote by $\Gamma_1, \Gamma_2, \ldots, \Gamma_R$ the ordinary functional principal components matrices of the sample curves associated with the functional predictors $\{X_1(t), X_2(t), \ldots, X_R(t) : t \in T\}$, respectively, the functional principal component logit model in terms of the logit transformations is expressed in matrix form as

$$L = \mathbf{1}\alpha + \Gamma_1\gamma_1 + \Gamma_2\gamma_2 + \cdots + \Gamma_R\gamma_R + U_1\delta_1 + \cdots + U_S\delta_S,$$

where $(\mathbf{1} \mid \Gamma_1 \mid \Gamma_2 \mid \cdots \mid \Gamma_R \mid U_1 \mid \cdots \mid U_S)$ is the design matrix in terms of ordinary functional principal components, $\gamma_r = \left(\gamma_{r1}, \ldots, \gamma_{rp_r}\right)'$ are the coefficients of the multiple model associated to the corresponding functional principal components and $(\delta_1, \ldots, \delta_s)'$ the scalar parameters associated to non-functional variables. By using a reduced set of $q_1, q_2, \ldots, q_R$ functional principal components, being the scores matrix denoted as $\Gamma_{1(q_1)}, \Gamma_{2(q_2)}, \ldots, \Gamma_{R(q_R)}$, respectively, the model is then expressed as

$$L = \mathbf{1}\alpha + \Gamma_{1(q_1)}\gamma_1 + \Gamma_{2(q_2)}\gamma_2 + \cdots + \Gamma_{R(q_R)}\gamma_R + U_1\delta_1 + \cdots + U_S\delta_S.$$

Basis coefficients for each functional parameter are then obtained by formula (11) from their corresponding γ parameter and the intercept α by formula (10).

`logitFD.pc` is the function from **logitFD** package that fits the ordinary functional principal component logit regression model. The declaration of the function has this form:

```
logitFD.pc(Response,FDobj=list(),ncomp=c(),nonFDvars=NULL),
```

and the function arguments are the following:

- `Response`: vector of responses $y_1, \ldots, y_n$.
- `FDobj`: list of the different functional objects (`fdobj`) to use from the `fd` package. Theoretically $x_1(t), \ldots, x_R(t)$.
- `ncomp`: vector with the number of functional principal components q to use in the model for each functional predictor. The length of the vector must be equal to the length of the `FDobj` list. The first element of the vector corresponds with the number of functional principal components of the first functional predictor (columns of Γ_1), the second with the columns of Γ_2, ..., the Rth with the columns of Γ_R.
- `nonFDvars`: data frame with the observations of the scalar predictor variables, that is, with columns $U_1 \ldots, U_S$. Let us observe that the number of rows of this data frame must be the same

as the length of the response vector. Likewise, the number of functions in each functional object must be the same for all functional objects.

In order to illustrate the performance of the function, let us consider `StationsVar$North` as a binary response variable, `TempMonth` and `PrecMonth` as functional predictors, and `StationsVar[,c("altitude","longitude")]` as scalar predictor variables. We are going to consider the first 3 and 4 functional principal components of `TempMonth` and `PrecMonth` respectively.

Our fit is obtained as

```
Fit1 <- logitFD.pc(Response=StationsVars$North,FDobj=list(TempMonth.fd,PrecMonth.fd),
ncomp = c(3,4),nonFDvars = StationsVars[,c("altitude","longitude")])
```

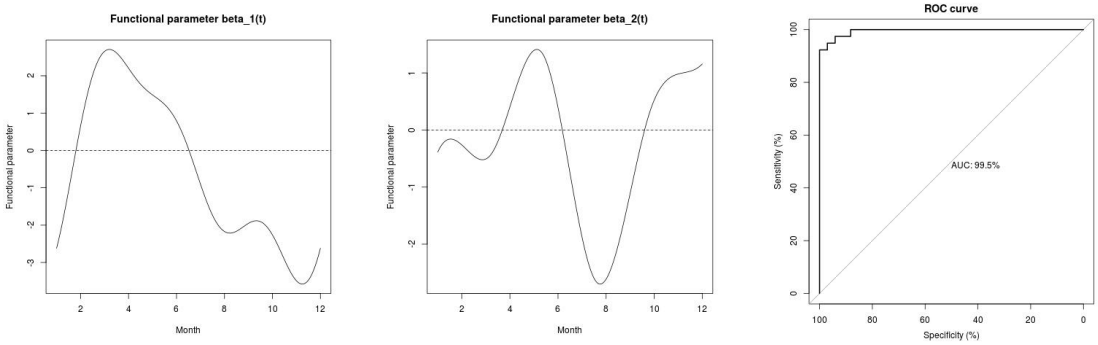
The output of the function is an `R` list with objects: `glm.fit`, `Intercept`, `betalist`, `PC.variance` and `ROC.curve`. These elements are explained next.

`glm.fit` **object of** `Fit1`**:** Object of class inherited from `"glm"`. This object contains details about the fit of the multiple logit model to explain the binary response from the selected functional principal components and the scalar variables. This output allows to use different R functions as `summary()` function to obtain or print a summary of the fit, or `anova()` function to produce an analysis of variance table, and to extract various useful features of the values returned by `"glm"` as *coefficients*, *effects*, *fitted.values* or *residuals* (see R help). In our example the summary of the fit can be seen on page (241). Let us observe that the package assigns the names `A.1`, `A.2` and `A.3` and `B.1`, `B.2`, `B.3` and `B.3` to the first 3 and 4 functional principal components of the functional covariates. From this object it would easily be able to make an analysis of residuals, with `residuals()` function, or fitted values, with `fitted.values()` function, testing goodness of fit, etc. A classical goodness of fit measure is the correct classification rate (CCR) from the classification table. In our example both elements can be easily obtained through these sentences `table(StationsVars$North,round(predict(Fit1$glm.fit,type = "response")))` and `100*sum(diag(table(StationsVars$North,round(predict(Fit1$glm.fit,type = "response"))))/nrow(StationsVars)`. From the results we can conclude that if we want to model the weather stations location from the temporal evolution of temperatures and precipitation and from altitude and longitude variables, we classify correctly 94.5% of stations.

`Intercept` **object of** `Fit1`**:** The α (intercept) estimated parameter through expression (10) is given in the object `Fit1$Intercept`.

`betalist` **object of** `Fit1`**:** A list of functional objects. Each element of the list contains the functional parameter corresponding to the associated functional predictor variable located in the same position of `FDobj` parameter that appears in the function. In our case, firsty temperature curves where introduced, and precipitation curves were added in second place. Then the first two elements of `betalist`, that is, `[[1]]` and `[[2]]` will be the functional parameters associated with temperature and precipitation curves respectively. If we use more functional data, `[[3]]`,`[[4]]`,... provide the corresponding functional parameters. Let us remember that as `fdobj`, its elements are `coefs`: the matrix (vector in this case) of basis coefficients, `basis`: the same basis used in `FDobj` object and the rest elements as `fdnames`. Besides, multiple functions from `fd` package can be used such as the `plot()` function, used here as `plot(Fit1$betalist[[1]])` and `plot(Fit1$betalist[[2]])` for the parameter functions associated to Temperature and Precipitation curves respectively. The plots that generate these sentences can be seen in Figure 2. We could also evaluate these functions in a grid with the function `eval.fd()`, for example in the observed months-time, we could obtain the values on page (241).

`PC.variance` **object of** `Fit1`**:** A list of *data.frames* with explained variability of functional principal components. Each element of the list contains the cumulative variance matrix corresponding to the associated functional variable in the same position. In our case, the first input curves were temperature curves and the second ones, the precipitation curves. In this point, the first element `[[1]]` of `PC.variance` will be the matrix of explained variability of functional principal components associated with temperature curves whereas the second element `[[2]]` of the `PC.variance` will be the matrix of explained variability of principal components associated with precipitation curves as with `betalist`. If we use more functional data `[[3]]`,`[[4]]`,... the function provides the corresponding explained variability matrices. The output got in `PC.variance` list is on page (242). We can observe that the first two functional principal component of temperature and precipitation accumulate 99.4% and 99.1% of the total variability respectively, so that the selection of 3 components for temperature and 4 for precipitation are enough for a good representation of the curves.



$\hat{\beta}_1(t)$ $\hat{\beta}_2(t)$ ROC curve

Figure 2: Left and middle curves are the estimated functional parameters associated with Temperature and Precipitation functional data respectively of Fit 1. The graph on the right is the ROC Curve of the fit. These graphs are obtained from Fit 1 by means of `logitFD.pc()` function.

`ROC.curve` **object of** `Fit1`**:** an object of the `roc()` function from **pROC** package whose mission is to test the prediction ability of the model. This function builds a ROC curve and returns a `roc` object, i.e. a list of class `roc`. This object can be printed, plotted, or passed to many other functions (see reference manual). As default this element returns the area under the ROC curve with the object `Fit1$ROC`. The plot of the ROC curve with sentence `plot(Fit1$ROC)` can be seen in Figure 2. As it was stated from the correct classification rate, the ROC curve and its graph allows us to observe that the fit is accurate for this modeling.

4 Filtered functional principal components included in the model according to their explained variability

Alternatively to ordinary functional principal component logit regression, Escabias et al. (2014) discussed a different approach based on equivalences proved by Ocaña et al. (2007) and Ocaña et al. (1999) between different functional principal component analysis. These equivalences stated that given $x_1(t),\ldots,x_n(t)$ a set of curves, the functional principal component analysis of the transformed curves $L(x_1(t)),\ldots,L(x_n(t))$ defined by $L(x_i(t))=\sum_{j=1}^{p} a_{ij}^{*}\phi_j(t)$, being $(a_{i1}^{*},\ldots,a_{ip}^{*})'=\Psi^{1/2}(a_{i1},\ldots,a_{ip})'$, is equivalent to multivariate PCA of the design matrix $A\Psi$ associated with the functional logit model. In this expansion, the principal component curves $f_j^{*}(t)$ are expressed in terms of the basis functions as

$$f_j^{*}(t)=\sum_{k=1}^{p} F_{jk}^{*}\phi_k(t),\ j=1,\ldots,p,$$

where basis coefficients in matrix form are obtained as $F^{*}=\Psi^{-1/2}V$, being V the eigenvectors of the covariance matrix of $A\Psi$.

The original curves can be also approximated

$$L(x_i(t))=L(\overline{x}(t))+\sum_{j=1}^{p}\xi_{ij}^{*}f_j^{*}(t),\ i=1,\ldots,n,$$

where ξ_{ij}^{*} are the functional principal components scores of the transformed curves $L(x_1(t)),\ldots,L(x_n(t))$.

Now again the original curves can be approximated by using a reduced set of these functional principal components

$$L(x_i(t))\simeq L(\overline{x}(t))+\sum_{j=1}^{q<p}\xi_{ij}^{*}f_j^{*}(t),\ i=1,\ldots,n.$$

In order to avoid multicollinearity in functional logit model an alternative is to use filtered principal components (see Escabias et al., 2004). So let $x_1(t),\ldots,x_n(t)$ be a set of curves with mean curve $\overline{x}(t)$ and $y_1,\ldots,y_n$ the response associated observations. Let $\Gamma^{*}=(\xi_{ij}^{*})$ be the $n\times p$ matrix of functional principal components, and $f_1^{*}(t),\ldots,f_p^{*}(t)$ the principal component curves. The filtered functional

principal component logit regression can be expressed

$$\begin{aligned} l_i &= \alpha + \int_T \left(\overline{x}(t) + \sum_{j=1}^{p} \xi_{ij}^* f_j^*(t) \right) \beta(t)\, dt = \alpha + \int_T \overline{x}(t)\beta(t)\, dt + \sum_{j=1}^{p} \xi_{ij}^* \int_T f_j^*(t)\, \beta(t)\, dt, \\ &= \gamma_0^* + \sum_{j=1}^{p} \xi_{ij}^* \gamma_j^*,\ i = 1, \ldots, n. \end{aligned}$$

This expression also allows expressing the basis coefficients of the functional parameter and the intercept parameter of the logit model alternatively in terms of the parameters estimated from the filtered functional principal components of the curves equivalently to (11) and (10) respectively:

$$\alpha = \gamma_0^* - \int_T \overline{x}(t)\beta(t)\, dt = \alpha + (\overline{a}_1, \ldots, \overline{a}_p)\Psi(\beta_1, \ldots \beta_p)', \tag{12}$$

$$(\beta_1, \ldots, \beta_p)' = \Psi F^*(\gamma_1^*, \ldots, \gamma_p^*)', \tag{13}$$

due to F^* and Ψ matrices are orthogonal and Ψ is also a symmetric matrix.

If we consider a principal component expansion of curves in terms of a reduced set of filtered functional principal components we can get an estimation of the basis coefficients of the functional parameter whose accuracy depends of the accumulated variability of the selected principal components (see Escabias et al., 2004).

So, if we denote by $\Gamma_1^*, \Gamma_2^*, \ldots, \Gamma_R^*$ the matrices of filtered functional principal components of the sample curves of the functional predictors $\{X_1(t), X_2(t), \ldots, X_R(t) : t \in T\}$ respectively, the functional principal component logit model in terms of the logit transformations is expressed in matrix form as

$$L = \mathbf{1}\alpha + \Gamma_1^*\gamma_1^* + \Gamma_2^*\gamma_2^* + \cdots + \Gamma_R^*\gamma_R^* + U_1\delta_1 + \cdots + U_S\delta_S,$$

where $\left(\mathbf{1} \mid \Gamma_1^* \mid \Gamma_2^* \mid \cdots \mid \Gamma_R^* \middle| U_1 \middle| \cdots \middle| U_S\right)$ is the design matrix in terms of ordinary functional principal components, $\gamma_r^* = \left(\gamma_{r1}^*, \ldots, \gamma_{rp_r}^*\right)'$ are the coefficients of the multiple model associated to the corresponding filtered functional principal components and $(\delta_1, \ldots, \delta_s)'$ the scalar parameters associated to non-functional variables. By using a reduced set of $q_1, q_2, \ldots, q_R$ filtered functional principal components $\Gamma_{1(q_1)}^*, \Gamma_{2(q_2)}^*, \ldots, \Gamma_{R(q_R)}^*$, respectively, the model is then expressed as

$$L = \mathbf{1}\alpha + \Gamma_{1(q_1)}^*\gamma_1^* + \Gamma_{2(q_2)}^*\gamma_2^* + \cdots + \Gamma_{R(q_R)}^*\gamma_R^* + U_1\delta_1 + \cdots + U_S\delta_S.$$

Basis coefficients for each functional parameter are then obtained by formula (11) from their corresponding γ^* parameters and the Intercept α^* by formula (10).

The function of the `logitFD` package that allows fitting the filtered functional principal components logit regression model is `logitFD.fpc`. The performance of the function is the same as the `logitFD.pc` function.

In order to illustrate the performance of the functions, let us again consider `StationsVar$North` as binary response variable, `TempMonth` and `PrecMonth` as functional predictors, and as scalar predictor variables `StationsVar[,c("altitude","longitude")]`. We are going to consider the first 3 and 4 functional principal components of `TempMonth` and `PrecMonth`, respectively.

Our fit is obtained as

```
Fit2 <- logitFD.fpc(Response=StationsVars$North,FDobj=list(TempMonth.fd,PrecMonth.fd),
ncomp = c(3,4),nonFDvars = StationsVars[,c("altitude","longitude")])
```

The output of this function is an `R` list with the same elements that were explained in the previous section. Next, the results of the fit are shown.

`glm.fit` **object of** `Fit2`**:** explained in the previous section, its results can be seen next to the ones obtained for `Fit1`

```
summary(Fit1$glm.fit)

Call:
glm(formula = design, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.77059  -0.01185   0.00000   0.01309   2.02115

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 15.10398    8.80373   1.716   0.0862 .
A.1         -1.94776    1.05278  -1.850   0.0643 .
A.2         -0.19686    0.58414  -0.337   0.7361
A.3         -6.69297    3.49893  -1.913   0.0558 .
B.1          0.41633    0.78514   0.530   0.5959
B.2          0.51503    6.42736   0.080   0.9361
B.3         -3.11044    3.06542  -1.015   0.3103
B.4         -2.44083    5.69108  -0.429   0.6680
altitude    -0.02846    0.01576  -1.806   0.0709 .
longitude    1.40203    0.85922   1.632   0.1027
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 100.857  on 72  degrees of freedom
Residual deviance:  14.785  on 63  degrees of freedom
AIC: 34.785

Number of Fisher Scoring iterations: 15
```

```
summary(Fit2$glm.fit)

Call:
glm(formula = design, family = binomial)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.671e-04  -2.100e-08   2.100e-08   2.100e-08   2.939e-04

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    598.163  75918.975   0.008    0.994
A.1            -99.485  11468.867  -0.009    0.993
A.2            -13.281  10608.315  -0.001    0.999
A.3           -264.950  45230.675  -0.006    0.995
B.1             26.123   7055.585   0.004    0.997
B.2            174.667  31244.941   0.006    0.996
B.3            318.127  79802.859   0.004    0.997
B.4           -828.247 233825.008  -0.004    0.997
altitude        -1.251    142.820  -0.009    0.993
longitude       50.865   7090.131   0.007    0.994
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0086e+02  on 72  degrees of freedom
Residual deviance: 2.2821e-07  on 63  degrees of freedom
AIC: 20

Number of Fisher Scoring iterations: 25
```

Classification table of `Fit2`: obtained as explained for `Fit1`, we can observe that the filtered functional principal components provide a better fit with CCR of 100% in spite a less accurate estimation of parameters due to the high standard error of coefficients estimation. This fact was observed and stated in Aguilera et al. (2008a) for example.

`Intercept` **object of** `Fit2`**:** provides the same result seen in the `Fit1` case.

`betalist` **object of** `Fit2`**:** The functional parameters obtained by this fit can be seen in Figure 3. We can observe the great similarity of the functional parameters form provided by the fit in terms of ordinary functional principal components and in terms of filtered functional principal components. The evaluation of these functions in the observed months-time, appears next with the ones obtained for `Fit1`:

`Fit1`

	Months	Beta1	Beta2
1	Jan	-2.6186728	-0.3861178
2	Feb	0.6541213	-0.2615488
3	Mar	2.6530440	-0.5074478
4	Apr	2.2007300	0.4053187
5	May	1.4871676	1.3946294
6	Jun	0.7918409	0.3944050
7	Jul	-0.9034488	-1.8776571
8	Aug	-2.1700722	-2.6123256
9	Sep	-1.9520934	-1.1094367
10	Oct	-2.2614869	0.5243271
11	Nov	-3.4888495	0.9816193
12	Dec	-2.6186728	1.1577407

`Fit2`

	Months	Beta1	Beta2
1	Jan	-114.45858	-208.85221
2	Feb	16.52657	-295.08973
3	Mar	97.13723	-142.07106
4	Apr	80.32782	29.22866
5	May	54.09383	91.48928
6	Jun	29.28698	-28.35074
7	Jul	-36.52664	-219.37744
8	Aug	-87.77921	-234.26248
9	Sep	-81.81846	-18.66271
10	Oct	-97.27569	226.44071
11	Nov	-148.43166	303.70498
12	Dec	-114.45858	61.99109

The code used for generating these results is `data.frame("Months" = names(monthLetters),"Beta1" = eval.fd(c(1:12),Fit1$betalist[[1]]),"Beta2" = eval.fd(c(1:12),Fit1$betalist[[2]]))` for the left-hand side and changing `Fit1` by `Fit2` for the right-hand side.

`PC.variance` **object of** `Fit2`**:** it can be observed that there are several differences in the dynamic of variance accumulation between ordinary and filtered functional principal component analysis.

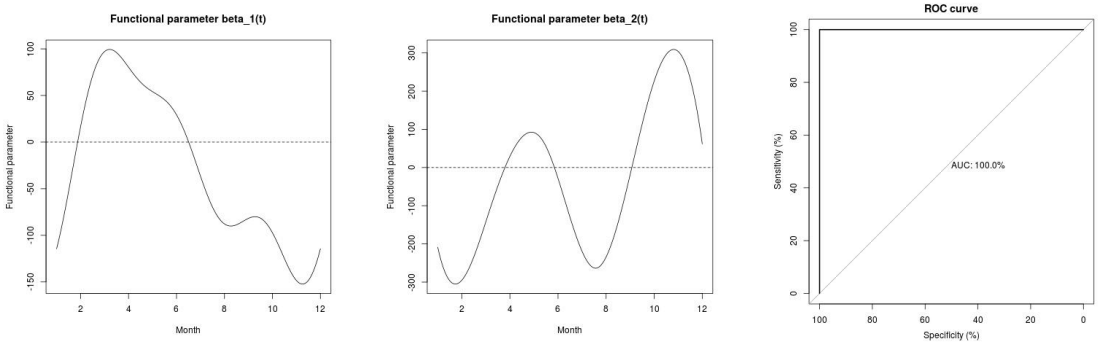


Figure 3: Left and middle curves are the estimated functional parameters associated with Temperature and Precipitation functional data respectively of Fit 2. The graph on the right is the ROC Curve of the fit. These graphs are obtained from Fit 1 by means of `logitFD.fpc()` function.

```
---------------------------------
Fit1$PC.variance
[[1]]
  Comp. % Prop.Var % Cum.Prop.Var
1   A.1        85.9           85.9
2   A.2        13.5           99.4
3   A.3         0.4           99.8
4   A.4         0.1           99.9
5   A.5         0.0           99.9
6   A.6         0.0           99.9
7   A.7         0.0           99.9
[[2]]
  Comp. % Prop.Var % Cum.Prop.Var
1   B.1        98.2           98.2
2   B.2         0.9           99.1
3   B.3         0.6           99.7
4   B.4         0.3          100.0
5   B.5         0.1          100.1
6   B.6         0.0          100.1
7   B.7         0.0          100.1
8   B.8         0.0          100.1
---------------------------------
```

```
---------------------------------
Fit2$PC.variance
[[1]]
  Comp. % Prop.Var % Cum.Prop.Var
1   A.1     85.888          85.888
2   A.2     13.479          99.367
3   A.3      0.440          99.807
4   A.4      0.132          99.939
5   A.5      0.034          99.973
6   A.6      0.016          99.989
7   A.7      0.010          99.999
[[2]]
  Comp. % Prop.Var % Cum.Prop.Var
1   B.1     99.070          99.070
2   B.2      0.536          99.606
3   B.3      0.311          99.917
4   B.4      0.049          99.966
5   B.5      0.031          99.997
6   B.6      0.002          99.999
7   B.7      0.000          99.999
8   B.8      0.000          99.999
---------------------------------
```

`ROC.curve` **object of** `Fit2`**:** explained in the previous Section, the plot of the ROC curve appears Figure 3. This graph and the area under the ROC curve (100%) show an improvement of the prediction ability of the fit with filtered functional principal components in comparison with ordinary functional principal components.

5 Ordinary and filtered functional principal components included in the model according to their prediction ability (stepwise method)

Escabias et al. (2004) proposed two alternative methods to include functional principal components in the logit model for both FPCA types: ordinary or filtered. On the one hand, functional principal components would be able to be included in the model in the order given by their explained variability. In that case the user should decide the number of functional principal components to include in the model for getting an accurate estimation of the functional parameter or for getting good prediction ability for the response. On the other hand, an automatic selection method of functional principal components could be used by a stepwise method. In this case the prediction ability of functional principal components would be the criterium to select the functional principal components and data would be the responsible of the model fit and prediction.

logitFD package contains two functions to fit the functional logit model after a stepwise selection procedure of functional principal components (ordinary and filtered) and nonfunctional variables. The fits obtained by these stepwise procedures are shown next.

```
Fit3 <- logitFD.pc.step(Response=StationsVars$North,FDobj=list(TempMonth.fd,PrecMonth.fd),
nonFDvars = StationsVars[,c("altitude","longitude")])
Fit4 <- logitFD.fpc.step(Response=StationsVars$North,FDobj=list(TempMonth.fd,PrecMonth.fd),
nonFDvars = StationsVars[,c("altitude","longitude")])
```

Let us observe that for these functions it is not necessary to use a number of components parameter in the functions. We call `Fit3` for ordinary functional principal component analysis and `Fit4` for filtered functional principal component analysis.

The output of these function are R lists with the same elements as the ones seen in `Fit1` and `Fit2`. We only show and explain here some of the results.

`glm.fit` **objects of** `Fit3` **and** `Fit4`**:** We can observe from these results that stepwise method selected three functional principal components for Temperature and only one for Precipitation. Regarding scalar predictors, the method selected the altitude variable. Note that stepwise selection included the same components for both approaches, although the values of their parameters and standard errors are different. The classification ability of these fits is 100% of correct classification rate and can be obtained by using the same code shown fof `Fit1` and `Fit2`.

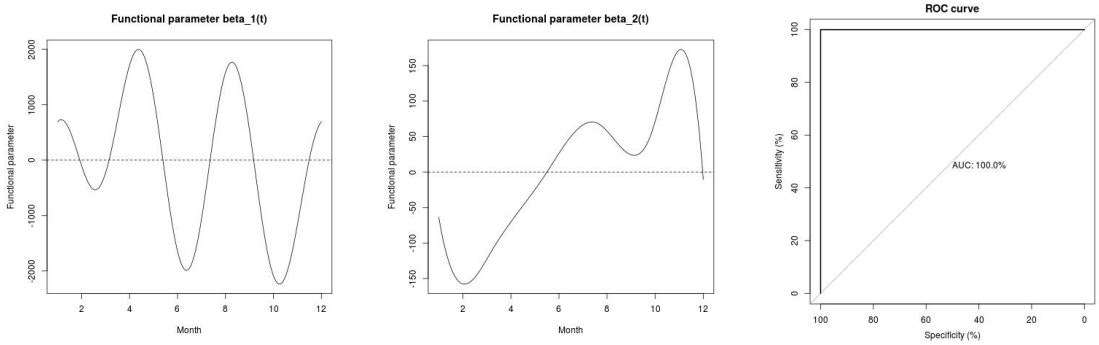
```
summary(Fit3$glm.fit)

Call:
glm(formula = Response ~ A.1 + altitude + A.7 + A.3 + B.5,
family = binomial, data = design)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-3.677e-04  -2.000e-08   2.000e-08   2.000e-08   2.960e-04

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   936.784  71065.432   0.013    0.989
A.1          -223.554  16658.601  -0.013    0.989
altitude       -2.543    191.207  -0.013    0.989
A.7          4016.721 300525.378   0.013    0.989
A.3          -972.450  73148.168  -0.013    0.989
B.5           308.717  23326.153   0.013    0.989

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0086e+02  on 72  degrees of freedom
Residual deviance: 4.7820e-07  on 67  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

```
summary(Fit4$glm.fit)

Call:
glm(formula = Response ~ A.1 + altitude + A.7 + A.3 + B.5,
family = binomial, data = design)

Deviance Residuals:
       Min         1Q     Median         3Q        Max
-6.974e-04  -2.000e-08   2.000e-08   2.000e-08   5.753e-04

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.938e+03  3.312e+05   0.006    0.995
A.1         -3.899e+02  3.754e+04  -0.010    0.992
altitude    -4.731e+00  6.218e+02  -0.008    0.994
A.7          6.724e+03  1.132e+06   0.006    0.995
A.3         -1.557e+03  8.121e+04  -0.019    0.985
B.5          6.409e+02  6.659e+04   0.010    0.992

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0086e+02  on 72  degrees of freedom
Residual deviance: 1.1402e-06  on 67  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

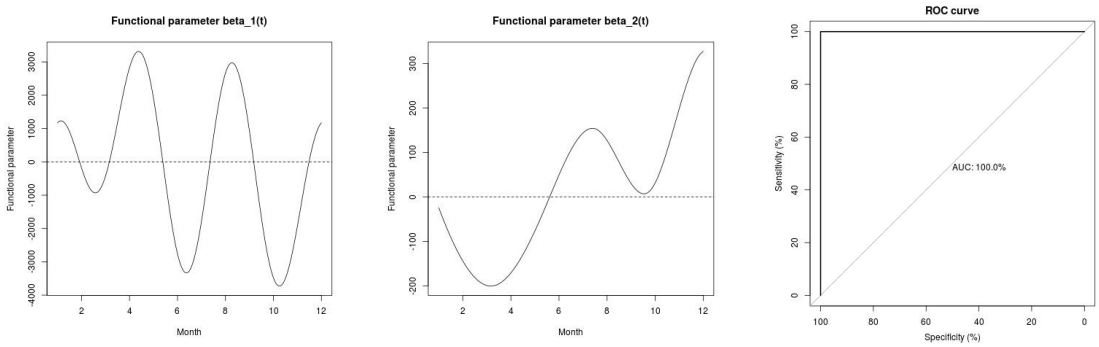
`betalist` **objects of** `Fit3` **and** `Fit4`**:** The graphs of estimated functional parameters are shown in Figure 4. It can be seen the similarity in the forms of the functional parameters, in spite of the evaluation values are different as can be seen next:

```
   Months  Fit3.Beta1 Fit3.Beta2 Fit4.Beta1 Fit4.Beta2
1     Jan   693.10831  -63.47274  1172.6949  -24.28034
2     Feb   -98.76707 -157.45404  -186.0346 -144.18907
3     Mar  -229.11217 -122.61836  -423.5395 -199.28285
4     Apr  1711.91853  -70.18329  2831.9461 -170.88030
5     May  1152.29655  -24.62758  1904.3853  -76.25583
6     Jun -1640.56224   26.48668 -2761.1827   49.39343
7     Jul -1066.07148   66.45460 -1780.0473  143.75270
8     Aug  1580.73125   57.60667  2663.1874  126.59096
9     Sep   543.54551   24.31157   921.5509   29.89969
10    Oct -2086.28319   71.63066 -3481.0951   31.57623
11    Nov -1163.57269  171.95001 -1925.9015  198.33040
12    Dec   693.10831  -10.48963  1172.6949  326.95671
```

The code used for thsese evaluations was similar as the one shown for Fit 1 and Fit 2: `data.frame("Months" = names(monthLetters),"Fit3.Beta1" = eval.fd(c(1:12),Fit3$betalist[[1]]),"Fit3.Beta2" = eval.fd(c(1:12),Fit3$betalist[[2]]),"Fit4.Beta1" = eval.fd(c(1:12),Fit4$betalist[[1]]),"Fit4.Beta2" = eval.fd(c(1:12),Fit4$betalist[[2]]))`



Ordinary FCPA and stepwise order



Filtered FCPA and stepwise order

Figure 4: Left and middle curves are the estimated functional parameters associated with Temperature and Precipitation functional data respectively of Fit 3 (up) and Fit 4 (down). The graphs on the right are the ROC Curve of the fits. These graphs are obtained from Fits 3 and 4 by means of `logitFD.pc.step()` and `logitFD.fpc.step()` functions respectively.

`PC.variance` **objects of** `Fit3` **and** `Fit4`**:** The objects of variance accumulation of the different functional principal components analysis do not change from the ones shown in previous sections. We do not show them here, the reader can check these equalities through the objects `Fit3$PC.variance` and `Fit4$PC.variance`.

`ROC.curve` **objects of** `Fit3` **and** `Fit4`**:** Roc objects with Roc areas provide an area under the roc curve of 100% in each case. The plot of the ROC curves showing the good performance of the fits can be seen in Figure 4.

6 Conclusions

In this work the functions of the **logitFD** package have been shown for fitting an extended functional principal components logistic regression model. The package provides two alternative solutions (ordinary and filtered FPCA) for the multicollinearity problem that arises when the functional predictors and the parameter functions are assumed to belong to the same finite dimensional space generated by a basis of functions. The dimension of the basis can be different in each functional variable in the model. Likewise, for each of the proposed solutions, two ways of choosing the functional principal components are provided: on the one hand, the users must manually choice the adequate number of components to be included in the model in order of variability, i.e., the first q principal components that overcome a certain variability percentage; on the other hand in the automatic order provided by the stepwise method, that is, according to predictive ability of principal components and non-functional variables.

The illustration of the use of the package's functionalities has been carried out using a set of functional and non-functional data, included in the **fda.usc** package. In particular, weather functional variables observed in 73 Spanish weather stations, such as the mean monthly evolution of temperatures and rainfall, and non-functional as the spatial location of the weather stations in the Spanish territory are considered throughout the current manuscript.

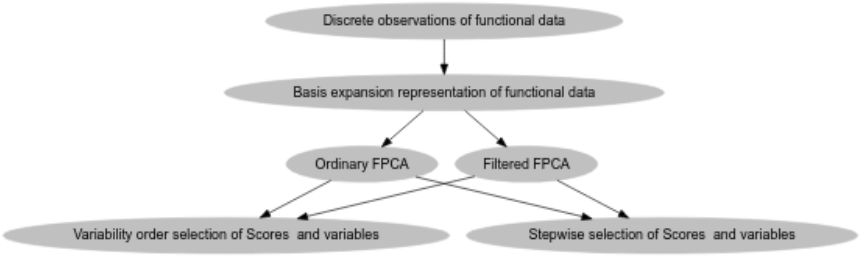


Figure 5: Diagram of steps for Functional Principal Components Logit Regression fit in its different situations considered in **logitFD** package: (1) basis expansion representation from discrete observations of curves, (2) choice the type of FPCA (ordinary or filtered) and (3) choice the method for scores selection (variability order or stepwise order).

The conclusions we have reached after the fits can be summarized in that the variables that best describe the North-South location of the meteorological stations are the mean monthly precipitation and temperature (through their first, third and seventh principal components for temperature and fifth for rainfall) and the own altitude of the weather stations. All the models provide good predictive ability, with the solutions based on ordinary and filtering FPCA by stepwise selection being the best (100 % CCR) due to their balance between reduced dimension and predictive ability. Likewise, the filtered FPCA solution including the components in order of variability provides results equally good to the previous ones but with more variables. The ordinary FPCA-based solution including the components in order of variability provides results similar to those previously described.

As was stated in the Introduction section, the `fregre.glm` function of the **fda.usc** package aim to achieve the same goal as the functions included in **logitFD** package, but through different point of view: `fregre.glm` use a discrete based methodology of functional data and **logitFD** functions use a purely functional approach using fd objects from the **fda** package. This approach makes the functional models of scalar response to suffer of multicollinearity problems with the inaccurate estimation of the functional parameters as a consequence (see Escabias et al., 2004). Two solutions based on functional PCA are implemented in **logitFD** package: (1) classic functional PCA and (2) filtered functional PCA. Each of PCA methods have been revealed to be useful in a different aspect: the first allow a lower estimation error of the basic coefficients of the functional parameters, while the second allow a lower estimation error of the proper curve, in terms of mean integrated quadratic error (see Escabias et al., 2004). Moreover the literature has also shown for methods involving principal components, that sometimes principal components with low variability explanation can be good predictors of the response, so a stepwise selection method of functional principal components has been included. So the main difference among **logitFD** functions and `fregre.glm` is that all the mentioned issues are addressed in the **logitFD** package and solved in a fast and transparent way and they are not taken into account in `fregre.glm`. Finally it is important to point out that the output of the functional elements of the **logitFD** functions (as the functional parameters) are also `fd` objects and therefore all the functions of the **fda** package could be used with them for plotting, evaluating, etc.

In short, logitFD package provides its users with the possibilities to deal with the functional logit regression model from basis expansion methodology of sample curves and solving in a fast and transparent way, the problems that arise through functional principal component analysis. In our opinion, if we wanted to solve the same problems by using alternative R functions with similar goal, it would be necessary give many steps that would make the process to be highly tedious. For this reason, and due to logit regression is highly considered in real problems, we think that the current manuscript can be very interesting for the readers given that they could use it as reference manual in their analysis

Figure 5 give a schematic diagram that summarize the steps of the methodology followed along the paper

7 Acknowledgment

This paper is partially supported by the project FQM-307 of the Government of Andalusia (Spain) and by the project PID2020-113961GB-I00 of the Spanish Ministry of Science and Innovation (also supported by the FEDER programme). They also acknowledge the financial support of the Consejería de Conocimiento, Investigación y Universidad, Junta de Andalucía (Spain) and the FEDER programme for project A-FQM-66-UGR20. Additionally, the authors acknowledge financial support by the IMAG–María de Maeztu grant CEX2020-001105-M/AEI/10.13039/501100011033.

Bibliography

C. Acal, A. M. Aguilera, and M. Escabias. New modeling approaches based on varimax rotation of functional principal components. *Mathematics*, 8(11):2085, 2020. URL https://doi.org/10.3390/math8112085. [p231]

A. M. Aguilera, M. Escabias, and M. J. Valderrama. Discussion of different logistic models with functional data. application to Systemic Lupus Erythematosus. *Computational Statistics and Data Analysis*, 53(1):151–163, 2008a. URL https://doi.org/10.1016/j.csda.2008.07.001. [p241]

A. M. Aguilera, M. Escabias, and M. J. Valderrama. Forecasting binary longitudinal data by a functional PC-ARIMA model. *Computational Statistics and Data Analysis*, 52(6):3187–3197, 2008b. URL https://doi.org/10.1016/j.csda.2007.09.015. [p231]

A. M. Aguilera, M. Escabias, C. Preda, and G. Saporta. Using basis expansion for estimating functional PLS regression. Applications with chemometric data. *Chemometrics and Intelligent Laboratory Systems*, 104(2):289–305, 2010. URL https://doi.org/10.1016/j.chemolab.2010.09.007. [p231]

A. M. Aguilera, M. C. Aguilera-Morillo, and C. Preda. Penalized versions of functional PLS regression. *Chemometrics and Intelligent Laboratory Systems*, 154:80–92, 2016. URL https://doi.org/10.1016/j.chemolab.2016.03.013. [p231]

M. C. Aguilera-Morillo and A. M. Aguilera. P-spline estimation of functional classification methods for improving the quality in the food industry. *Communications in Statistics - Simulation and Computation*, 44(10):2513–2534, 2015. URL https://doi.org/10.1080/03610918.2013.804555. [p231]

M. C. Aguilera-Morillo, A. M. Aguilera, M. Escabias, and M. J. Valderrama. Penalized spline approaches for functional logit regression. *TEST*, 22(2):251–277, 2013. URL https://doi.org/10.1007/s11749-012-0307-1. [p231]

A. Bianco, G. Boente, and G. Chebi. Penalized robust estimators in sparse logistic regression. *TEST*, in press:1–32, 2021. URL https://doi.org/10.1007/s11749-021-00792-w. [p231]

C. De Boor. *A practical guide to splines (revised edition)*. Springer, 2001. [p232]

M. Escabias, A. M. Aguilera, and M. J. Valderrama. Principal component estimation of functional logistic regression: discussion of two different approaches. *Journal of Nonparametric Statistics*, 16 (3-4):365–384, 2004. URL https://doi.org/10.1080/10485250310001624738. [p231, 235, 236, 237, 239, 240, 242, 245]

M. Escabias, A. M. Aguilera, and M. J. Valderrama. Modeling environmental data by functional principal component logistic regression. *Environmetrics*, 16(1):95–107, 2005. URL https://doi.org/10.1002/env.696. [p231, 232]

M. Escabias, A. M. Aguilera, and M. J. Valderrama. Functional (pls) logit regression model. *Computational Statistics and Data Analysis*, 51(10):4891–4902, 2007. URL https://doi.org/10.1016/j.csda.2006.08.011. [p232]

M. Escabias, M. J. Valderrama, A. M. Aguilera, M. E. Santofimia, and M. C. Aguilera-Morillo. Stepwise selection of functional covariates in forecasting peak levels of olive pollen. *Stochastic Environmental Research and Risk Assessment*, 27(2):367–376, 2013. URL https://doi.org/10.1007/s00477-012-0655-0. [p231]

M. Escabias, A. M. Aguilera, and M. C. Aguilera-Morillo. Functional PCA and base-line logit models. *Journal of Classification*, 31(3):296–324, 2014. URL https://doi.org/10.1007/s00357-014-9162-y. [p231, 239]

M. Febrero-Bande and M. Oviedo. Statistical computing in fucntional data analysis: The R package fda.usc. *Journal of statistical software*, 51(4):1–28, 2012. URL https://doi.org/10.18637/jss.v051.i04. [p231, 232]

F. Ferraty and P. Vieu. *Nonparametric functional data analysis. Theory and practice*. Springer-Verlag, 2006. [p231]

L. Horvath and P. Kokoszka. *Inference for functional data with applications*. Springer-Verlag, 2012. [p231]

D. Hosmer, S. Lemeshow, and R. Sturdivant. *Applied Logistic Regression*. Wiley, 2013. [p233]

G. M. James. Generalized linear models with functional predictors. *Journal of the Royal Statistical Society. Series B*, 64(3):411–432, 2002. URL https://doi.org/10.1111/1467-9868.00342. [p231]

P. Kokoszka and M. Reimherr. *Introduction to Functional Data Analysis*. CRC Press, 2018. [p231]

S. Mousavi and H. Sørensen. Functional logistic regression: a comparison of three methods. *Journal of Statistical Computation and Simulation*, 88(2):250–268, 2018. URL https://doi.org/10.1080/00949655.2017.1386664. [p231]

H. G. Müller and U. Stadtmüller. Generalized functional linear models. *Annals of Statistics*, 33(2): 774–805, 2005. URL https://doi.org/10.1214/009053604000001156. [p231]

F. A. Ocaña, A. M. Aguilera, and M. J. Valderrama. Functional principal components analysis by choice of norm. *Journal of Multivariate Analysis*, 71(2):262–276, 1999. URL https://doi.org/10.1006/jmva.1999.1844. [p239]

F. A. Ocaña, A. M. Aguilera, and M. Escabias. Computational considerations in functional principal component analysis. *Computational Statistics*, 22(3):449–465, 2007. URL https://doi.org/10.1007/s00180-007-0051-2. [p236, 239]

C. Preda and G. Saporta. PLS regression on a stochastic process. *Computational Statistics and Data Analysis*, 48(1):149–158, 2005. URL https://doi.org/10.1016/j.csda.2003.10.003. [p231]

J. O. Ramsay and B. W. Silverman. *Functional data analysis (Second Edition)*. Springer-Verlag, 2005. [p231, 232, 236]

J. O. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer-Verlag, 2009. [p231, 232]

A. Tapia, V. Leiva, M. Diaz, and V. Giampaoli. Influence diagnostics in mixed effects logistic regression models. *TEST*, 28:920–942, 2019. URL https://doi.org/10.1007/s11749-018-0613-3. [p231]

M. Vidal, M. Rosso, and A. M. Aguilera. Bi-smoothed functional independent component analysis for eeg artifact removal. *Mathematics.*, 9(11):1243, 2021. URL https://doi.org/10.3390/math9111243. [p231]

J. Zhang. *Analysis of Variance for functional data*. CRC Press, 2014. [p231]

H. Zhu, F. Yao, and H. Zhang. Structured functional additive regression in reproducing kernel hilbert spaces. *Journal of the Royal Statistical Society B*, 76(3):581–603, 2014. URL https://doi.org/10.1111/rssb.12036. [p231]

Manuel Escabias
Department of Statistics and Operation Research, University of Granada
Facultad de Farmacia, Campus de Cartuja. 18071 Granada
Spain
ORCiD: 0000-0002-1653-9022
escabias@ugr.es

Ana M. Aguilera
Department of Statistics and Operation Research, University of Granada
Facultad de Ciencias, Campus Fuentenueva. 18071 Granada
Spain
ORCiD: 0000-0003-2425-6716
aaguiler@ugr.es

Christian Acal
Department of Statistics and Operation Research, University of Granada
Facultad de Ciencias, Campus Fuentenueva. 18071 Granada
Spain
ORCiD: 0000-0002-2636-5396
chracal@ugr.es

eat: An R Package for fitting Efficiency Analysis Trees

by Miriam Esteve, Victor España, Juan Aparicio, and Xavier Barber

Abstract eat is a new package for R that includes functions to estimate production frontiers and technical efficiency measures through non-parametric techniques based upon regression trees. The package specifically implements the main algorithms associated with a recently introduced methodology for estimating the efficiency of a set of decision-making units in Economics and Engineering through Machine Learning techniques, called Efficiency Analysis Trees (Esteve et al. 2020). The package includes code for estimating input- and output-oriented radial measures, input- and output-oriented Russell measures, the directional distance function and the weighted additive model, plotting graphical representations of the production frontier by tree structures, and determining rankings of importance of input variables in the analysis. Additionally, it includes the code to perform an adaptation of Random Forest in estimating technical efficiency. This paper describes the methodology and implementation of the functions, and reports numerical results using a real data base application.

1 Introduction

Efficiency analysis refers to the discipline of estimating production frontiers while measuring the efficiency of a set of observations, named Decision Making Units (DMUs), which use several inputs to produce several outputs. In the literature of Economics, Engineering and Operations Research, the estimation of production frontiers is a current topic of interest (see, for example, Arnaboldi, Azzone, and Giorgino 2014; Aparicio et al. 2017; O'Donnell et al. 2018). In this line, many models for estimating production frontiers have been developed, resorting to parametric and non-parametric approaches. In the non-parametric approach, a functional form does not need to be specified (e.g. a Cobb-Douglas production function) through the specification of a set of parameters to be estimated, since they are usually data-driven. Additionally, non-parametric models innately cope with multiple-output scenarios. In contrast, the parametric approach aggregates the outputs into a single production index or attempts to model the technology using a dual cost function (Orea and Zof 'io 2019). These are some of the advantages that makes the non-parametric approaches for measuring technical efficiency more appealing than their parametric counterparts.

Contextualizing the non-parametric measurement of efficiency analysis requires outlining the following works. Farrel (1957) was a renowned opponent of estimating efficiency by determining average performance and, indeed, he was the first author in the literature to introduce a method for constructing production frontiers as the maximum producible output from an input bundle. Inspired by Koopmans (1951) and Debreu (1951), Farrell introduced a piece-wise linear upper enveloping surface of the data cloud as the specification of the production frontier, satisfying some microeconomics postulates: free disposability, convexity and minimal extrapolation. A DMU is considered technically inefficient if it is located below the frontier. Furthermore, Farrell's measure of efficiency, inspired by Shephard (1953), is based on radial movements (equiproportional changes) from technically inefficient observations to their benchmarks located at the estimated production frontier. In the same context as Farrell, Afriat (1972) determined a production frontier under non-decreasing and concavity mathematical assumptions and, at the same time, as close as possible to the sample of observations. Finally, in the same line of research, Charnes, Cooper, and Rhodes (1978) and Banker, Charnes, and Cooper (1984) proposed Data Envelopment Analysis (DEA), rooted in mathematical programming to provide a relative efficiency assessment of a set of DMUs by the construction of a piece-wise linear frontier. Along with DEA, Free Disposal Hull (FDH) is another of the most recognized non-parametric models for estimating production frontiers. FDH is a deterministic model introduced by Deprins and Simar (1984), which is only based on the free disposability and minimal extrapolation principles, as opposed to DEA, which also assumes convexity. In fact, FDH can be considered the skeleton of DEA, since the convex hull of the former coincides with DEA's frontier (see Daraio and Simar 2005). In addition, other recent alternative non-parametric techniques for estimating production frontiers are: Banker and Maindiratta (1992) and Banker (1993), who showed that DEA can be interpreted as a Maximum Likelihood estimator; Simar and Wilson (1998, 2000; 2000), who introduced how to determine confidence intervals for the efficiency score of each DMU through adapting the bootstrapping methodology by Efron (1979) to the context of FDH and DEA; or Kuosmanen and Johnson (2010; 2017), who have recently shown that DEA may be interpreted as non-parametric least-squares regression, subject to shape constraints on the production frontier and sign constraints on residuals; to name a few.

However, few of the above methodologies are based upon Machine Learning techniques, despite being a rising research field (see, for example, the recent papers by Khezrimotlagh et al. 2019; and Zhu et al. 2019; or the book by Charles, Aparicio, and Zhu 2020). Recently, a bridge has been built between these literatures, Machine Learning and production theory, through a new technique proposed in Esteve et al. (2020), called Efficiency Analysis Trees. This new method shares some similarities with the standard FDH technique. In contrast to FDH, Efficiency Analysis Trees overcomes the well-known problem of overfitting linked to FDH and DEA, by using cross-validation to prune back the deep tree obtained in a first stage. Esteve et al. (2020) also showed that the performance of Efficiency Analysis Trees, checked through Monte Carlo simulations, clearly outperforms the FDH technique with respect to bias and mean squared error.

Many of the standard models for estimating technical efficiency are nowadays available as R packages such as: **Benchmarking** (Bogetoft and Otto 2010), for estimating technologies and measuring efficiencies using Data Envelopment Analysis (DEA) and Stochastic Frontier Analysis (SFA); **nonparaeff** (Oh and Suh 2013), for measuring efficiency and productivity using DEA and its variations; **npbr** (Daouia, Laurent, and Noh 2017), which covers data envelopment techniques based on piece-wise polynomials, splines, local linear fitting, extreme values and kernel smoothing; **snfa** (McKenzie 2018), which fits both a smooth analogue of DEA and a non-parametric analogue of SFA; or **semsfa** (Ferrara and Vidoli 2018), which, in a first stage, estimates Stochastic Frontier Models by semiparametric or non-parametric regression techniques to relax parametric restrictions and, in a second stage, applies a technique based on pseudolikelihood or the method of moments for estimating variance parameters. Additionally, there are other packages on efficiency measurement developed for alternative platforms. In MATLAB (The MathWorks Inc. 2021), we can find the **Data Envelopment Analysis Toolbox** (Álvarez, Barbero, and Zofio 2020), which implements the main DEA models and solves measures like the directional distance function (with desirable and undesirable outputs), the weighted additive model, and the Malmquist-Luenberger index; or the **Total Factor Productivity Toolbox** (Balk, Barbero, and Zof 'Io 2018), which includes functions to calculate the main Total Factor Productivity indices and their decomposition by DEA models. In Stata (StataCorp 2021), it is possible to find a similar package in Ji and Lee (2010).

In this paper, we introduce a new package in R, called **eat**, for fitting regression trees to estimate production frontiers in microeconomics and engineering, by implementing the main features of Efficiency Analysis Trees (Esteve et al. 2020). In particular, **eat** includes a complete set of baseline functions, covering a wide range of efficiency models fitted by Efficiency Analysis Trees (Esteve et al. 2020) and Random Forest (Esteve et al. 2021), and reporting numerical and graphical results. **eat** is available as free software, under the GNU General Public License version 3, and can be downloaded from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=eat`, including supplementary material as datasets or vignettes to replicate all the results presented in this paper. In addition, **eat** is hosted on an open source repository on GitHub at `https://github.com/MiriamEsteve/EAT`. The main objective of this package is the estimation of a production frontier through regression trees satisfying the microeconomic principles of free disposability, convexity and deterministic data. Free disposability states that if a certain input and output bundle is producible, then any input and output bundle that presents a greater value for inputs and a lower value for outputs is also producible. In some sense, it means that doing it worse is always feasible. Convexity means that if two input-output bundles are assumed producible, then any convex combination of them are also feasible. Finally, the deterministic quality means that the observations that belong to the data sample have been observed without noise. In other words, the technology always contains all these observations and, graphically, the production frontier envelops all the data cloud from above.

The efficiency measurement field has witnessed the introduction of many different technical efficiency measures throughout the last decades. Regarding the technical efficiency measures implemented in the new **eat** package, it is worth mentioning that numerical scores and barplots are provided for the output-oriented and input-oriented BCC radial models (Banker, Charnes, and Cooper 1984), the directional distance function (Chambers, Chung, and Färe 1998), the weighted additive model (Lovell and Pastor 1995; and W. W. Cooper, Park, and Pastor 1999) and the output-oriented and input-oriented Russell measures (Färe and Lovell 1978). Additionally, the adaptation of Random Forest (Breiman 2001) for dealing with ensembles of Efficiency Analysis Trees, recently introduced in Esteve et al. (2021) and denoted as RF+EAT, has been also incorporated into the new **eat** package. The frontier estimator based on Random Forest, which is associated with more robust results, also allows to determine out-of-sample efficiency evaluation for the assessed DMUs. Another remarkable aspect of Efficiency Analysis Trees is the inherited ability to calculate feature importance as performed by other tree-based models of Machine Learning. Specifically, this fact allows the researchers to know which are the most relevant variables for obtaining efficiency and thus getting an explanation of the level of technical efficiency identified for each assessed unit. This ranking of importance variable has been implemented in the **eat** package. Finally, and from a data visualization point of view, the obtained frontier from Efficiency Analysis Trees can be represented by means of a tree structure, ideal for high-dimensional

scenarios where the patterns between the inputs and the efficient levels of outputs are very complex. In addition, FDH and DEA standard models have been included in the new package in order to facilitate comparison with the efficiency scores determined by the Efficiency Analysis Trees technique. Also, the convexification of the estimation of the technology provided by Efficiency Analysis Trees, named Convexified Efficiency Analysis Trees (CEAT) by Aparicio et al. (2021), is implemented in the **eat** package, with the objective of determining estimations under the axiom of convexity.

The functions included in the **eat** package are summarized in Table 1. This table comprises two columns divided into four subsections for Efficiency Analysis Trees, Random Forest for Efficiency Analysis Trees, Convexified Efficiency Analysis Trees and functions for data simulation. The first column is the name of the main functions and the second one is the description of the functions and the reference of the paper in which we can find the most detailed theoretical explanation of the corresponding function.

The paper is organized as follows. The following section summarises the two methodologies implemented in the **eat** package in R: Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees. Section Data Structure describes the data structures that characterize the production possibility sets, the structure of the functions, the results, etc., and briefly explains which data are used to illustrate the package. Section Basic functions of the library presents the basic methods. The next Section Basic EAT and RFEAT models deals with the measurement of economic efficiency of FDH, DEA, Efficiency Analysis Trees, Random Forest for Efficiency Analysis Trees and Convexified Efficiency Analysis Trees models. Advanced options, including displaying and exporting results can be found in Section Advanced options and displaying and exporting results. Section Conclusions concludes.

Table 1: eat package functions.

Function	Description
Subsection 1: Efficiency Analysis Trees	
`EAT`	It generates a pruned Efficiency Analysis Trees model and returns an `EAT` object.
`bestEAT`	It computes the root mean squared error (RMSE) for a set of Efficiency Analysis Trees models made up of a set of user-entered hyperparameters. These models are fitted with a training sample and evaluated with a test sample.
`efficiencyEAT`	It computes the efficiency scores of a set of DMUs through an Efficiency Analysis Trees model and returns a `data.frame`. The FDH scores can also be computed. Alternative mathematical programming models for calculating the efficiency scores are: • `"BCC.OUT"`: The output-oriented BCC radial model. • `"BCC.INP"`: The input-oriented BCC radial model. • `"DDF"`: The directional distance function. • `"RSL.OUT"`: The output-oriented Russell model. • `"RSL.INP"`: The input-oriented Russell model. • `"WAM.MIP"`: The weighted additive model with Measure of Inefficiency Proportion. • `"WAM.RAM"`: The weighted additive model with Range Adjusted Measure of Inefficiency.
`efficiencyJitter`	It returns a jitter plot (from `ggplot2`) that represents the dispersion of the efficiency scores of the set of DMUs in the leaf nodes of an Efficiency Analysis Trees model. Mean and standard deviation of scores are shown.
`efficiencyDensity`	It returns a density plot (from `ggplot2`) to compare the distribution of efficiency scores between two given models (`"EAT"`, `"FDH"`, `"CEAT"`, `"DEA"` and `"RFEAT"` are available).

Continued on next page

Table 1 – continued from previous page

Function	Description
`plotEAT`	It returns a plot of the tree-structure (from `ggparty` and `partykit`) of an Efficiency Analysis Trees model.
`frontier`	It returns a plot (from `ggplot2`) of the estimated production function obtained by an Efficiency Analysis Trees model in a two-dimensional scenario (1 input and 1 output). Optionally, the FDH frontier can be plotted.
`predict`	Generic function to predict the expected output by an `EAT` object. The result is a `data.frame` with the predicted values.
`rankingEAT`	It returns a `data.frame` with the scores of variable importance obtained by an Efficiency Analysis Trees model and optionally a barplot representing the variable importance.
Subsection 2: Random Forest for Efficiency Analysis Trees	
`RFEAT`	It generates a Random Forest for Efficiency Analysis Trees model and returns an `RFEAT` object.
`bestRFEAT`	It computes the root mean squared error (RMSE) for a set of Random Forest for Efficiency Analysis Trees models made up of a set of user-entered hyper-parameters. These models are fitted with a training sample and evaluated with a test sample.
`efficiencyRFEAT`	It computes the efficiency scores of a set of DMUs through a Random Forest for Efficiency Analysis Trees model and returns a `data.frame`. The FDH scores can also be computed. Only the output-oriented BCC radial model is available.
`plotRFEAT`	It returns a line plot (from `ggplot2`) with the Out-of-Bag (OOB) error for a random forest consisting of k trees.
`predict`	Generic function to predict the expected output by an `RFEAT` object. The result is a `data.frame` with the predicted values.
`rankingRFEAT`	It returns a `data.frame` with the scores of variable importance obtained by a Random Forest for Efficiency Analysis Trees model and optionally a barplot representing the variable importance.
Subsection 3: Convexified Efficiency Analysis Trees	
`efficiencyCEAT`	It computes the efficiency scores of a set of DMUs through a Convexified Efficiency Analysis Trees model and returns a `data.frame`. The DEA scores can also be computed. Alternative mathematical programming models for calculating the efficiency scores are: • `"BCC.OUT"`: The output-oriented BCC radial model. • `"BCC.INP"`: The input-oriented BCC radial model. • `"DDF"`: The directional distance function. • `"RSL.OUT"`: The output-oriented Russell model. • `"RSL.INP"`: The input-oriented Russell model. • `"WAM.MIP"`: The weighted additive model with Measure of Inefficiency Proportion. • `"WAM.RAM"`: The weighted additive model with Range Adjusted Measure of Inefficiency.
Subsection 4: Functions for data simulation	
`Y1.sim`	It returns a `data.frame` with simulated data in a single output scenario (1, 3, 6, 9, 12 and 15 inputs can be generated).

Continued on next page

Table 1 – continued from previous page

Function	Description
`X2Y2.sim`	It returns a `data.frame` with simulated data in a scenario with 2 inputs and 2 outputs.

2 Background

Efficiency Analysis Trees

In this section, we briefly introduce the main fundaments of Efficiency Analysis Trees. Nevertheless, we first need to introduce some notation related to the standard Free Disposal Hull (FDH) and Classification and Regression Trees (CART) techniques.

We consider the observation of n Decision Making Units (DMUs), which consumes $\mathbf{x}_i = (x_{1i}, ..., x_{mi}) \in R_+^m$ quantity of inputs for the production of $\mathbf{y}_i = (y_{1i}, ..., y_{si}) \in R_+^s$ quantity of outputs[1]. The dataset is denoted in a compact way as $\aleph = \{(\mathbf{x}, \mathbf{y})\}_{i=1,...,n}$. The so-called production possibility set or technology, which is the set of technically feasible combinations of $(\mathbf{x}, \mathbf{y})$, is defined, in general terms, as:

$$\psi = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \mathbf{x} \textit{ can produce } \mathbf{y}\right\}. \tag{1}$$

On this set, certain assumptions are usually made (see, Färe and Primont 1995), such as: monotonicity (free disposability) of inputs and outputs, which means that if $(\mathbf{x}, \mathbf{y}) \in \psi$, then $(\mathbf{x'}, \mathbf{y'}) \in \psi$, as long as $\mathbf{x'} \geq \mathbf{x}$ and $\mathbf{y'} \leq \mathbf{y}$; and convexity, i.e., if $(\mathbf{x}, \mathbf{y}) \in \psi$ and $(\mathbf{x'}, \mathbf{y'}) \in \psi$, then $\lambda (\mathbf{x}, \mathbf{y}) + (1 - \lambda) (\mathbf{x'}, \mathbf{y'}) \in \psi$, $\forall \lambda \in [0, 1]$. In the case of the FDH estimator, only free disposability is assumed. Additionally, FDH is assumed to be deterministic. In other words, the production possibility set determined by FDH always contains all the observations that belong to the data sample and, graphically, the production frontier envelops the data cloud from above. Also, FDH satisfies the minimal extrapolation postulate, which is associated with the typical problem-solving principle of Occam's razor. That is, additional requirements are needed to select the right estimator because there are a lot of possible estimators that can meet free disposability and the deterministic quality. In this sense, according to Occam's razor, the most conservative estimate of the production frontier would be that related to a surface that would envelop the data from above, satisfy free disposability and, at the same time, be as close as possible to the data cloud. In contrast, the DEA estimator requires stronger assumptions, such as convexity of the set ψ.

With regard to the measurement of technical efficiency, a certain part of the set ψ is actually of interest. It is the efficient frontier or production frontier of ψ, which is defined as $\partial(\boldsymbol{\psi}) := \{(\mathbf{x}, \mathbf{y}) \in \boldsymbol{\psi} : \hat{\mathbf{x}} < \mathbf{x}, \hat{\mathbf{y}} > \mathbf{y} \Rightarrow (\hat{\mathbf{x}}, \hat{\mathbf{y}}) \notin \boldsymbol{\psi}\}$. Technical efficiency is understood as the distance from a point belonging to ψ to the production frontier $\partial(\boldsymbol{\psi})$. In particular, Deprins and Simar (1984) proposed the FDH estimator of the set ψ from the dataset $\aleph$ as:

$$\hat{\boldsymbol{\psi}}_{FDH} = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \exists i = 1, ..., n \text{ such that } \mathbf{y} \leq \mathbf{y}_i, \mathbf{x} \geq \mathbf{x}_i\right\}. \tag{2}$$

The FDH technique is very attractive because it is based on very few suppositions, but it suffers from overfitting due to its construction. This problem is shared by other well-known data-based approaches. For example, Classification and Regression Trees (CART), a technique that belongs to the field of machine learning, suffer problems of overfitting when a deep tree is developed. However, this problem can be fixed using a cross-validation process to prune the deep tree. The principle behind CART is relatively simple: a certain criterion is chosen to recursively generate binary partitions of the data until a meaningful division is no longer possible or a stopping rule is maintained. The graphic result of this approach is a tree that starts at the root node, develops through the intermediate nodes and ends at the terminal nodes, also known as leaves. The binary nature of CART is represented by each parent node, except for the leaves, giving rise to two child nodes.

Next, we briefly introduce the recent technique named Efficiency Analysis Trees by Esteve et al. (2020). This technique allows the estimation of production frontiers, fulfilling the common axioms of microeconomics, through a data-based approach that is not founded on any particular distribution on the data noise and, in addition, creates a step function as a estimator. It shares these characteristics with the FDH technique, but the overfitting problem related to FDH can be solved through cross-validation based on pruning.

We now introduce the main steps of the algorithm linked to the Efficiency Analysis Trees technique.

[1]We use bold for denoting vectors, and non-bold for scalars.

Let us assume that we have a node t in the tree structure to be split. This node contents a subset of the original sample $\aleph$. The algorithm has to select an input variable j, $j = 1, ..., m$, and a threshold $s_j \in S_j$, where S_j is the set of possible thresholds for variable j, such that the sum of the mean squared error (MSE) calculated for the data that belong to the left child node t_L and the MSE corresponding to the data belonging to the right child node t_R is minimized. The data of the left child node t_L satisfies the condition $x_j < s_j$, while the data of the right child node t_R satisfies the condition $x_j \geq s_j$. Additionally, in the algorithm, the set S_j is defined from the observed values of the input j in the data sample $\aleph$. Formally, the split consists in selecting the combination $\left(x_j, s_j\right)$ which minimizes $R\left(t_L\right) + R\left(t_R\right) = \frac{1}{n}\sum_{(x_i,y_i)\in t_L}\sum_{r=1}^{s}\left(y_{ri} - y_r\left(t_L\right)\right)^2 + \frac{1}{n}\sum_{(x_i,y_i)\in t_R}\sum_{r=1}^{s}\left(y_{ri} - y_r\left(t_R\right)\right)^2$, where $y_r\left(t\right)$ denotes the estimation of the r-th output of the node t. One of the most important aspects in the production context is how to define $y_r\left(t\right)$ in each node for fulfilling the free disposability property during the growing process of the tree. In this sense, the notion of Pareto-dominance between nodes introduced in Esteve et al. (2020) is really relevant.

As described above, each node t is defined by a series of conditions in the input space as $\left\{x_j < s_j\right\}$ or $\left\{x_j \geq s_j\right\}$. In this sense, after executing the split, a region in the input space is created. This region in the input space is called the "support" of node t and is defined as $\text{supp}\left(t\right) = \left\{\mathbf{x} \in R_+^m : a_j^t \leq x_j < b_j^t, j = 1, ..., m\right\}$. The parameters a_j^t and b_j^t are originated from the several thresholds selected during the splitting process. Giving the notion of support of a node, it is possible to establish the concept of Pareto-dominance. Let $k = 1, ..., K$ be the total number of splits executed. Let $T_k\left(\aleph\right)$ be the tree built after the k-th split. Let $\tilde{T}_k\left(\aleph\right)$ be the set of leaves in the tree $T_k\left(\aleph\right)$. More notation: let $t^* \in \tilde{T}_k\left(\aleph\right)$ be the node to be split in a certain step of the algorithm, then $T\left(k|t^* \to t_L, t_R\right)$ denotes the tree associated with this specific split. Let $k = 1, ..., K$ and $t \in \tilde{T}_k\left(\aleph\right)$, then the set of Pareto-dominant nodes of node t is defined as $P_{T_k(\aleph)}\left(t\right) = \{t' \in \tilde{T}_k\left(\aleph\right) - t : \exists \mathbf{x} \in \text{supp}\left(t\right), \exists \mathbf{x'} \in \text{supp}\left(t'\right) \text{ such that } \mathbf{x'} \leq \mathbf{x}\}$. $P_{T_k(\aleph)}\left(t\right)$ contains all the nodes such that at least one input vector in its corresponding support, non-necessarily observed, dominates at least one input vector belonging to the support of node t (in the Pareto sense). To do so, in practice, it is only necessary to compare the components of $\mathbf{a}^{t'}$ and $\mathbf{b}^t$. Specifically, $\mathbf{a}^{t'} < \mathbf{b}^t$ if and only if $t' \in P_{T_k(\aleph)}\left(t\right)$.

Now, we return to how to estimate the outputs in each child node with the aim of guaranteeing the satisfaction of free disposability. For any node $t^* \in \tilde{T}_k\left(\aleph\right)$, the way to estimate the value of the outputs for the right child node is through the estimation of the outputs of its parent node, i.e., $y_r\left(t_R\right) = y_r(t^*),\ r = 1, ..., s$, while the estimation of outputs for the left child node is:

$$y_r\left(t_L\right) = \max\left\{\max\left\{y_{ri} : \left(\mathbf{x}_i, \mathbf{y}_i\right) \in t_L\right\}, y_r\left(I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right)\right\},\ r = 1, ..., s, \tag{3}$$

where $y_r\left(I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right) = \max\left\{y_r\left(t'\right) : t' \in I_{T(k|t^*\to t_L,t_R)}\left(t_L\right)\right\}$ and $y_r\left(t'\right)$ is the estimation of the output y_r at node $t' \in \tilde{T}\left(k|t^* \to t_L, t_R\right)$, $r = 1, ..., s$. This way of estimating the output values guarantees free disposability.

Accordingly, the algorithm selects the best pair $\left(x_{j^*}, s_{j^*}\right)$ such that the sum of the MSE of the left and right child nodes is minimized. Once the split of node t^* is executed, the tree $T\left(k|t^* \to t_L^*, t_R^*\right)$ is obtained. This process continues until bipartition is not possible because all the data in a node have the same input values or a certain stopping rule is satisfied. The usual stopping rule is $n\left(t\right) \leq n_{\min} = 5$, where $n(t)$ is the sample size of node t. The final tree built is denoted as $T_{\max}\left(\aleph\right)$, which usually is a deep tree.

$T_{\max}\left(\aleph\right)$ suffers from the same problem as FDH, i.e., overfitting. Esteve et al. (2020) proposed to prune the tree exploiting the same technique as Breiman et al. (1984). This pruning process resorts to the notion of the error-complexity measure $R_\alpha\left(T\left(\aleph\right)\right)$, which is a combination between a measure of the accuracy of the tree, defined as the sum of the MSE determined at each leaf node, and a measure of the number of leaf nodes. Also, $R_\alpha\left(T\left(\aleph\right)\right)$ depends on a parameter α, which compensates the values of the two components of the error: $R_\alpha\left(T\left(\aleph\right)\right) = R\left(T\left(\aleph\right)\right) + \alpha\left|\tilde{T}\left(\aleph\right)\right|$. The idea behind the pruning of $T_{\max}\left(\aleph\right)$ is to minimize $R_\alpha\left(T\left(\aleph\right)\right)$. The pruning process is also based on cross-validation (see Breiman et al. (1984) for more details). The tree resulting from the pruned process is $T^*\left(\aleph\right)$. This tree doesn't suffer from the overfitting problem. For this reason, the use of $T^*\left(\aleph\right)$ is recommended rather than $T_{\max}\left(\aleph\right)$, unless a descriptive analysis of the sample is required.

Finally, $\mathbf{d}_{T^*(\aleph)}\left(\mathbf{x}\right)$ will denote hereinafter the multi-dimensional estimator defined from $T^*\left(\aleph\right)$ and the sample $\aleph$, i.e., $d_{rT^*(\aleph)}\left(\mathbf{x}\right) = \sum_{t\in T^*(\aleph)} y_r\left(t\right) I\left(\mathbf{x} \in t\right)$, for all $r = 1, ..., s$, with $I\left(\cdot\right)$ being the indication function. From this estimator, it is possible to define a production possibility set or technology estimated from the Efficiency Analysis Trees technique as:

$$\hat{\Psi}_{T^*(\aleph)} = \left\{\left(\mathbf{x}, \mathbf{y}\right) \in R_+^{m+s} : \mathbf{y} \leq \mathbf{d}_{T^*(\aleph)}\left(\mathbf{x}\right)\right\}. \tag{4}$$

$\hat{\Psi}_{T^*(\aleph)}$ satisfies free disposability and the deterministic quality.

By analogy with the existing relationship between FDH and DEA, it is possible to derive an estimation of Ψ by the convexification of the set $\hat{\Psi}_{T^*}$. In this sense, the convexification of the production possibility set derived from EAT would be as follows:

$$conv\left(\hat{\Psi}_{T^*}\right) = \left\{(\mathbf{x},\mathbf{y}) \in R_+^{m+s} : \mathbf{x} \geq \sum_{t\in\tilde{T}^*} \lambda_t \mathbf{a}^t, \mathbf{y} \leq \sum_{t\in\tilde{T}^*} \lambda_t \mathbf{d}_{T^*}\left(\mathbf{a}^t\right), \sum_{t\in\tilde{T}^*} \lambda_t = 1, \boldsymbol{\lambda} \geq 0_{|\tilde{T}^*|}\right\}. \quad (5)$$

Under the convexity assumption, the EAT methodology is known as the Convexified Efficiency Analysis Trees technique (hereinafter referred to as CEAT) (see Aparicio et al. 2021).

Random Forest for Efficiency Analysis Trees

In this section, we briefly describe the extension of the approach by Esteve et al. (2020) to the context of using ensembles of trees to provide estimates of production frontiers (see Esteve et al. 2021). Specifically, we briefly revise the way to adapt the standard Random Forest (Breiman 2001) for estimating production frontiers satisfying fundamental postulates of microeconomics, such as free disposability. The adaptation of Random Forest to the estimation of production frontiers by Esteve et al. (2021) is the first one that focuses on the introduction of a methodology for measuring technical efficiency that is robust to the resampling of data and, at the same time, to the specification of input variables.

Data robustness and resampling methods for input modeling are both topics of interest in the literature on technical efficiency measurement. Regarding robustness to data, Simar and Wilson (1998, 2000; 2000) were the first ones to adapt the bootstrapping methodology (Efron 1979) to the context of DEA and FDH. As regards the importance of the robustness of input and output variables in non-parametric efficiency analysis, since the beginning of DEA and FDH, researchers have always been aware that the selection of input and output variables to be considered in efficiency analysis is one of the crucial issues in the specification of the model. In practice, the researchers' previous experience may lead to the selection of some inputs and outputs considered essential to represent the underlying technology. However, there may be other variables whose inclusion in the model the analyst is not always sure of (Pastor, Ruiz, and Sirvent 2002). Some approaches focus on balancing the experience of researchers with the information provided by observations (see, for example, Banker 1993, 1996; Pastor, Ruiz, and Sirvent 2002). Another recent approach is based, in contrast, on determining efficiency scores that are robust to variable selection by considering all the possible combinations of inputs and outputs and their aggregation (Landete, Monge, and Ruiz 2017).

On the whole, Random Forest (Breiman 2001) is an ensemble learning method that works by constructing a multitude of decision trees by CART (Breiman et al. 1984) at training time and aggregating the information of the individual trees in a final prediction value. In particular, when Random Forest is applied to regression problems, the final estimator corresponds to the mean of each individual prediction (Breiman 2001). Random Forest modifies the growing process of an individual tree as follows, by: (i) applying bootstrapping on the data training for each individual tree and (ii) selecting a random subset of the predictors in each iteration. In this way, given a learning sample $\aleph$ of size n, Random Forest repeatedly selects random samples of size n with replacement of the set $\aleph$. Then, the method fits the trees to these samples but, to do this, it uses a modified tree learning algorithm that chooses, in each candidate division of the learning process, a random subset of predictors. The reason for doing this is due to the instability of the model. It is known that individual decision trees, such as CART, are very unstable (Berk 2016). This means that completely different tree structures are given when the training data is modified slightly. In this way, the result of applying Random Forest is an estimator that overcomes overfitting and instability problems in general, resulting in a substantial reduction in variance.

The algorithm associated with the adaptation of the Random Forest technique to the world of technical efficiency assessment, called RF+EAT, is introduced in Esteve et al. (2021). The steps that must be carried out in Random Forest for Efficiency Analysis Trees are shown in Algorithm 1. This algorithm is based on the typical algorithm of Random Forest that can be found in Kuhn, Johnson, et al. (2013). In Algorithm 1, the first step consists of selecting the number of trees that will make up the forest, that is, the hyperparameter p. Then, p (bootstrap) random samples from the original data sample with replacement are generated. Next, the Efficiency Analysis Trees algorithm by Esteve et al. (2020) is applied to each subsample applying the stopping rule $n(t) \leq n_{\min}$, but without pruning. Also, in this algorithm, $n_{\min}$ is treated as an additional hyperparameter that could be tuned. During the execution of the Efficiency Analysis Trees algorithm, a subset of input variables (*mtry*) from the original set is randomly selected each time the splitting subroutine is applied. To do that, one of the

following five thumb rules is used following the literature:

- Breiman's Rule: $mtry = \frac{m}{3}$,
- Rule DEA1: $mtry = \frac{n(t)}{2} - s$ (Golany and Roll 1989; Homburg 2001),
- Rule DEA2: $mtry = \frac{n(t)}{3} - s$ (Nunamaker 1985; Banker et al. 1989; Friedman and Sinuany Stern 1998; Raab and Lichty 2002),
- Rule DEA3: $mtry = \frac{n(t)}{2s}$ (Dyson et al. 2001),
- Rule DEA4: $mtry = \min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$ (W. Cooper et al. 2007).

Input: p, number of trees
$\aleph$, original data
Output: $\{T(\aleph_q) : q = 1, ..., p\}$
for $q = 1$ ***to*** p **do**
  $\aleph_q$:= Bootstrap sample of $\aleph$
  $T(\aleph_q)$:= Efficiency Analysis Tree trained on $\aleph_q$
  foreach *split* **do**
    Randomly in $T(\aleph_q)$ selects $mtry(\leq m)$ of the original inputs using a specific rule;
    Select the best input in $T(\aleph_q)$ among the *mtry* inputs and split the data
  end
  $T(\aleph_q)$ is completed when $n(t) \leq n_{min}, \forall t$ leaf node of $T(\aleph_q)$
  ($T(\aleph_q)$ is not pruned)
end

Algorithm 1: Random Forest for Efficiency Analysis Trees algorithm for estimating production frontiers

Once Algorithm 1 has been applied, p fitted trees are determined with the aim of obtaining an output estimation giving an input vector $\mathbf{x} \in R_+^m$. In this regard, we have $T(\aleph_1), ..., T(\aleph_p)$ tree structures derived from the application of the Efficiency Analysis Trees algorithm on the p bootstrap subsamples $\aleph_1, ..., \aleph_p$. Given an input vector $\mathbf{x} \in R_+^m$, an output estimator is determined by averaging the individual estimator corresponding to each tree:

$$\mathbf{y}^{RF+EAT(\aleph)}(\mathbf{x}) := \frac{1}{p}\sum_{q=1}^{p}\mathbf{d}_{T(\aleph_q)}(\mathbf{x}). \tag{6}$$

where $\mathbf{d}_{T(\aleph_q)}(\mathbf{x})$ denotes the output estimator associated with each tree structure $T(\aleph_q)$, given an input vector $\mathbf{x} \in R_+^m$.

In addition, this estimator allows the technology or production possibility set to be defined as:

$$\hat{\Psi}_{RF+EAT} = \left\{(\mathbf{x}, \mathbf{y}) \in R_+^{m+s} : \mathbf{y} \leq \mathbf{y}^{RF+EAT(\aleph)}(\mathbf{x})\right\}. \tag{7}$$

As happens with the standard Random Forest, Random Forest for Efficiency Analysis Trees also exploits the Out-Of-Bag (OOB) concept. The OOB estimate at observation $(\mathbf{x}_i, \mathbf{y}_i)$ consists in evaluating the prediction of the ensemble just using the individual models $T(\aleph_q)$ whose corresponding bootstrap samples $\aleph_q$ are such that $(\mathbf{x}_i, \mathbf{y}_i) \notin \aleph_q$. From this definition, the generalization error is defined as the average of the OOB estimates calculated over all the observations in the learning sample $\aleph$:

$$err^{RF+EAT(\aleph)} = \frac{1}{n}\sum_{(x_i,y_i)\in\aleph}\sum_{r=1}^{s}\left(y_{ri} - y_r^{RF+EAT(\aleph)}(\mathbf{x}_i)\right)^2. \tag{8}$$

The generalization error is useful for determining a measure of variable importance, which can be used for creating a sorted list of inputs $x_1, ..., x_m$. The way to calculate the input importance of variable x_j is: firstly, generate a new database, $\aleph^j$, identical to the original one $\aleph$, where specifically the values of variable x_j were randomly permuted; secondly, apply Algorithm 1 on the new 'virtual' learning sample $\aleph^j$; thirdly, determine the value of the generalization error, i.e., $err^{RF+EAT(\aleph^j)}$; and, finally, calculate the percentage increase of the generalization error when variable x_j is shuffled as:

$$\%Inc^{RF+EAT}\left(x_j\right) = 100 \cdot \left(\frac{err^{RF+EAT\left(\aleph^j\right)} - err^{RF+EAT(\aleph)}}{err^{RF+EAT(\aleph)}} \right). \tag{9}$$

3 Data structure

Data are managed as a regular R `data.frame` in the **eat** functions (`matrix` is often accepted but will be converted to a `data.frame` in the functions pre-processing). The main functions of the **eat** package are `EAT()` and `RFEAT()`, which return structured objects named `EAT` and `RFEAT`, respectively. These objects contain fields with relevant information such as the estimation results or the arguments introduced by the user in the function call.

The fields of the `EAT` object are the following:

- `data`: Contains the input and output variables.
  - `df`: Data introduced by the user in a `data.frame` structure after being preprocessed.
  - `x`: Input indexes in `df`.
  - `y`: Output indexes in `df`.
  - `input_names`: Name of the input variables in `df`.
  - `output_names`: Name of the output variables in `df`.
  - `row_names`: Name of the observations in `df`.
- `control`: Contains the hyperparameters selected by the user.
  - `fold`: Number of folds in which is divided `df` to apply cross-validation.
  - `numStop`: Minimum number of observations in a node.
  - `max.leaves`: Maximum number of leaf nodes.
  - `max.depth`: Maximum number of nodes between the root node (not included) and the furthest leaf node.
  - `na.rm`: A logical variable that indicates if `NA` rows should be ignored.
- `tree`: `list` containing the nodes of the fitted Efficiency Analysis Trees model. Each node is made up of the following elements:
  - `id`: Node index
  - `F`: Father node index.
  - `SL`: Left child node index.
  - `SR`: Right child node index.
  - `index`: Set of indexes corresponding to the observations in a node.
  - `R`: Error at the node.
  - `xi`: Index of the variable that produces the split in a node.
  - `s`: Threshold of the variable `xi`.
  - `a`: The components of the vector $\mathbf{a}^t$.
  - `b`: The components of the vector $\mathbf{b}^t$.
- `nodes_df`: Contains the following information related to the nodes of the fitted Efficiency Analysis Trees model in a `data.frame` structure:
  - `id`: Node index
  - `N`: Number of observations in a node.
  - `Proportion`: Proportion of observations in a node.
  - `y`: Fitted values.
  - `R`: Error at the node.
  - `index`: Indexes of the observations in a node.
- `model`: Contains the following information related to the fitted Efficiency Analysis Trees model:
  - `nodes`: Number of nodes in the tree.
  - `leaf_nodes`: Number of leaf nodes in the tree.
  - `a`: The components of the vector $\mathbf{a}^t$.
  - `y`: Output estimation for each leaf node.

Regarding the `RFEAT` object, it contains the following fields:

- `data`: same fields as the `EAT` object.
- `control`: Contains the hyperparameters selected by the user.

  - numStop: Minimum number of observations in a node.
  - m: Number of trees that make up the random forest.
  - s_mtry: Number of inputs that can be randomly selected in each split.
  - na.rm: A logical variable that indicates if NA rows should be ignored.
- forest: A list containing the individual Efficiency Analysis Trees that make up the random forest.
- Error: The Out-of-Bag error at the random forest.
- OOB: A list containing the observations used for training each Efficiency Analysis Tree that makes up the random forest.

Dataset and statistical sources

```
# We load the library
library("eat")

# We load the data
data("PISAindex")
```

We illustrate all the models presented in this paper resorting to a single dataset (PISAindex) available in the **eat** package. Our dataset consists of 72 countries with 3 outputs and 13 inputs. The output data have been collected by the PISA (Programme for International Student Assessment) 2018 survey (OECD 2018) and refers to the average score in mathematics, reading and science domains for schools in these countries. Regarding the input data, the variables have been collected from the Social Progress Index (2018) and are related to the socioeconomic environment of these countries. These inputs can be classified into four blocks as follows:

- Basic Human Needs:
  - Nutrition and Basic Medical Care (NBMC)
  - Water and Sanitation (WS)
  - Shelter (S)
  - Personal Safety (PS).
- Foundations of Wellbeing:
  - Access to Basic Knowledge (ABK)
  - Access to Information and Communications (AIC)
  - Health and Wellness (HW)
  - Environmental Quality (EQ).
- Opportunity:
  - Personal Rights (PR)
  - Personal Freedom and Choice (PFC)
  - Inclusiveness (I)
  - Access to Advanced Education (AAE).
- Economy:
  - Gross Domestic Product based on Purchasing Power Parity (GDP_PPP).

Finally, in order to simplify the examples and reduce computation time, a subset of variables is selected as follows:

```
# Inputs (5): PR, PFC, I, AAE, GDP_PPP
# Outputs (3): S_PISA, R_PISA, M_PISA
PISAindex <- PISAindex[, c(3, 4, 5, 14, 15, 16, 17, 18)]

head(PISAindex)
```

```
#>     S_PISA R_PISA M_PISA    PR   PFC     I   AAE GDP_PPP
#> SGP    551    549    569 71.70 87.90 48.26 74.31  97.745
#> JPN    529    504    527 94.07 82.40 62.32 81.29  41.074
#> KOR    519    514    526 92.71 79.06 63.54 86.32  41.894
#> EST    530    523    523 95.67 84.10 55.58 73.16  35.308
#> NLD    503    485    519 96.34 89.04 75.82 82.99  56.455
#> POL    511    512    516 86.41 78.25 57.58 76.21  31.766
```

Table 2 reports the descriptive statistics for these variables (outputs and inputs).

Table 2: Descriptive statistics (averages, standard deviations, minimum, median and maximum) of input–output.

variable	type	mean	sd	min	median	max
S_PISA	output	455.06	48.32	336.00	466.00	551.00
R_PISA	output	450.89	50.52	340.00	466.00	549.00
M_PISA	output	454.81	52.17	325.00	463.50	569.00
PR	input	81.62	17.98	21.14	88.40	98.07
PFC	input	75.42	11.03	47.25	78.19	91.65
I	input	54.17	17.07	12.37	55.51	81.91
AAE	input	69.87	10.75	48.37	71.65	90.43
GDP_PPP	input	36.04	21.91	7.44	31.42	114.11

4 Basic functions of the library

In this section, we introduce the main functions of the library related to Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees. To execute the following examples, the package **eat** must be loaded and the seed 100 must be set for reproducibility.

```
# We set the seed
set.seed(100)
```

The EAT basic model

The basic model of Efficiency Analysis Trees that we explained in subsection Efficiency Analysis Trees can be implemented in R using the function `EAT()`:

```
EAT(
  data, x, y,
  numStop = 5,
  fold = 5,
  max.depth = NULL,
  max.leaves = NULL,
  na.rm = TRUE
)
```

The `EAT()` function is the cornerstone of the **eat** library. The minimum arguments of this function are the data (`data`) containing the study variables, the indexes of the predictor variables or inputs (`x`) and the indexes of the predicted variables or outputs (`y`). Additionally, the `numStop`, `fold`, `max.depth` and `max.leaves` arguments are included for more experienced users in the fields of machine learning and tree-based models. Modifying these four hyperparameters allows obtaining different frontier estimates and therefore selecting the one that best suits the needs of the analysis. The description of these parameters is as follows:

- `numStop` refers to the minimum number of observations in a node to be split and is directly related to the size of the tree. The higher the value of `numStop`, the smaller the size of the tree.
- `fold` refers to the number of parts in which the `data` is divided to apply the cross-validation technique. Variations in the `fold` argument are not directly related to the size of the tree.
- `max.depth` limits the number of nodes between the root node (not included) and the furthest leaf node. When this argument is introduced, the typical process of growth-pruning is not carried out. In this case, the tree is allowed to grow to the required depth.
- `max.leaves` determines the maximum number of leaf nodes. As in `max.depth`, the process of growth-pruning is not performed. In this respect, the tree grows until the required number of leaf nodes is reached, and then, the tree is returned.

Notice that including the arguments `max.depth` or `max.leaves` reduces the computation time by eliminating the pruning procedure. However, the pruning process is preferred if the objective of the study is inferential instead of simply descriptive. If both are included at the same time, a `warning` message is displayed and only `max.depth` is used.

As an example, using data from subsection Dataset and statistical sources, we next create a multi response tree using the suitable code as follows. Results are returned as an `EAT` object, as explained in Section Data structure.

```
modelEAT <- EAT(data = PISAindex, x = 4:8, y = 1:3)
```

The RFEAT basic model

The basic model of Random Forest for Efficiency Analysis Trees that we explained in subsection Random Forest for Efficiency Analysis Trees can be implemented in R using the function `RFEAT()`:

```
RFEAT(
  data, x, y,
  numStop = 5,
  m = 50,
  s_mtry = "BRM",
  na.rm = TRUE
)
```

The `RFEAT()` function has also been developed with the aim of providing greater statistical robustness to the results obtained by the `EAT()` function. The `RFEAT()` function requires the `data` containing the variables for the analysis, `x` and `y` corresponding to the inputs and outputs indexes respectively, the minimum number of observations in a node for a split to be attempted (`numStop`) and `na.rm` to ignore observations with `NA` cells. All these arguments are used for the construction of the p (this is denoted with `m` in the `RFEAT()` function) individual Efficiency Analysis Trees that make up the random forest. Finally, the argument `s_mtry` indicates the number of inputs that can be randomly selected in each split. It can be set as any integer although there are also certain predefined values. Let m be the number of inputs, let s be the number of outputs and let $n(t)$ be the number of observations in a node. Then, the predefined values for `s_mtry` are:

- `BRM` = $\frac{m}{3}$,
- `DEA1` = $\frac{n(t)}{2} - s$,
- `DEA2` = $\frac{n(t)}{3} - s$,
- `DEA3` = $\frac{n(t)}{2s}$,
- `DEA4` = $\min\left\{\frac{n(t)}{s}, \frac{n(t)}{3} - s\right\}$.

As an example, using data from subsection Dataset and statistical sources, we next create a forest with 30 trees. Results are returned as an `RFEAT` object, as explained in Section Data structure.

```
modelRFEAT <- RFEAT(data = PISAindex, x = 4:8, y = 1:3, m = 30)
```

Predictions

The estimators of the Efficiency Analysis Trees and Random Forest for Efficiency Analysis Trees can be computed in R using the function `predict()`:

```
predict(
  object,
  newdata,
  x, ...
)
```

Regarding the arguments of `predict()`, `object` can be an `EAT` or an `RFEAT` object, `newdata` refers to a `data.frame` and `x` to the set of inputs to be used. This function returns a `data.frame` with the expected output for a set of observations. For predictions using an `EAT` object, only one tree is used. However, for the RFEAT model, the output is predicted by each of the p (`m` in the `RFEAT()` function) individual trees trained and subsequently the mean value of all predictions is obtained.

As an example, we next evaluate the last 3 DMUs from the data of subsection Dataset and statistical sources and the corresponding EAT and RFEAT models. Results are returned in a `data.frame` structure with the output predictions:

```
predict(object = modelEAT, newdata = tail(PISAindex, 3), x = 4:8)
```

```
#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1         428         424         437
#> 2         377         359         368
#> 3         377         359         368
```

```
predict(object = modelRFEAT, newdata = tail(PISAindex, 3), x = 4:8)

#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1    439.9667    435.1333    441.2000
#> 2    402.0667    389.0667    403.9000
#> 3    399.0333    389.3333    399.6333
```

In the same way, the user can also create a new `data.frame` and calculate predictions for it as follows:

```
new <- data.frame(AAE = c(61, 72), PR = c(76, 81), I = c(41, 55), GDP_PPP = c(19, 31),
                  PFC = c(67, 78))

predict(object = modelEAT, newdata = new, x = 1:5)

#>   S_PISA_pred R_PISA_pred M_PISA_pred
#> 1         428         424         421
#> 2         481         479         488
```

Importance of predictor variables

The way to compute in R the predictor variables importance in the Efficiency Analysis Trees methodology is using the functions `rankingEAT()` or `rankingRFEAT()`:

```
# Through Efficiency Analysis Trees
rankingEAT(
  object,
  barplot = TRUE,
  threshold = 70,
  digits = 2
)

# Through Random Forest for Efficiency Analysis Trees
rankingRFEAT(
  object,
  barplot = TRUE,
  digits = 2
)
```

These functions allow a selection of variables by calculating a score of importance through Efficiency Analysis Trees or Random Forest for Efficiency Analysis Trees, respectively. These importance scores represent how influential each variable is in the model. Regarding the Efficiency Analysis Trees [`RankingEAT()`], the notion of surrogate splits by Breiman et al. (1984) was implemented. In this regard, the measure of importance of a variable x_j is defined as the sum over all nodes of the decrease in mean squared error produced by the best surrogate split on x_j at each node (see Definition 5.9 in Breiman et al. (1984)). Since only the relative magnitudes of these measures are interesting for researchers, the actual measures of importance that we report are normalized. In this way, the most important variable has always a value of 100, and the others are in the range 0 to 100. As for the Random Forest for Efficiency Analysis Trees [`RankingRFEAT()`], (9) was implemented for each input variable. Regarding the available arguments of the functions, the user can specify the number of decimal units (`digits`) and include a barplot (from `ggplot2`) with the scores of importance (`barplot`). Additionally, the `rankingEAT()` function allows to display a horizontal line in the graph to facilitate the cut-off point between important and irrelevant variables (`threshold`).

As an example, we next use the objects `modelEAT` (an `EAT` object from the `EAT()` function) and `modelRFEAT` (an `RFEAT` object from the `RFEAT()` function) created in the previous section to assess the predictors used. These functions return the name of the predictor variables, the scores of importance (in the range 0-100 for the `rankingEAT()` function) and a barplot (without horizontal line for the `rankingRFEAT()` function).

```
rankingEAT(object = modelEAT)

#> $scores
```

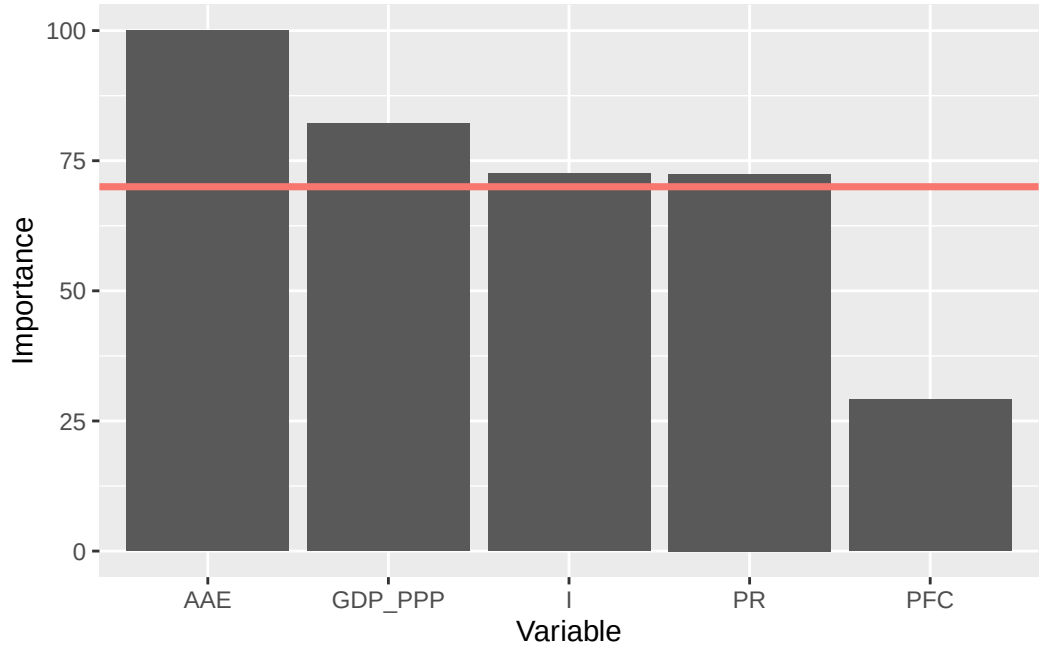


Figure 1: Barplot generated by applying 'rankingEAT()' to the PISAindex database to determine the ranking of variable importance.

```
#>         Importance
#> AAE         100.00
#> GDP_PPP      82.13
#> I            72.58
#> PR           72.45
#> PFC          29.07
#>
#> $barplot
```

```
rankingRFEAT(object = modelRFEAT)
```

```
#> $scores
#>         Importance
#> PR            1.75
#> PFC          -1.64
#> GDP_PPP      -2.16
#> I            -2.87
#> AAE          -3.67
#>
#> $barplot
```

Note that negative scores may appear when calculating the importance of variables using the `rankingRFEAT()` function. The appearance of this type of (negative) score can be understood as, if that variable were removed from the model, *ceteris paribus*, then an improvement in the predictive capacity of the model would be produced.

5 Basic EAT and RFEAT models

Efficiency scores are numerical values that indicate the degree of efficiency of a set of Decision Making Units (DMU). In the **eat** package, these scores can be calculated through an Efficiency Analysis Trees model, a Random Forest for Efficiency Analysis Trees model or a Convexified Efficiency Analysis Trees model. The code is as follows:

```
# For Efficiency Analysis Trees
```

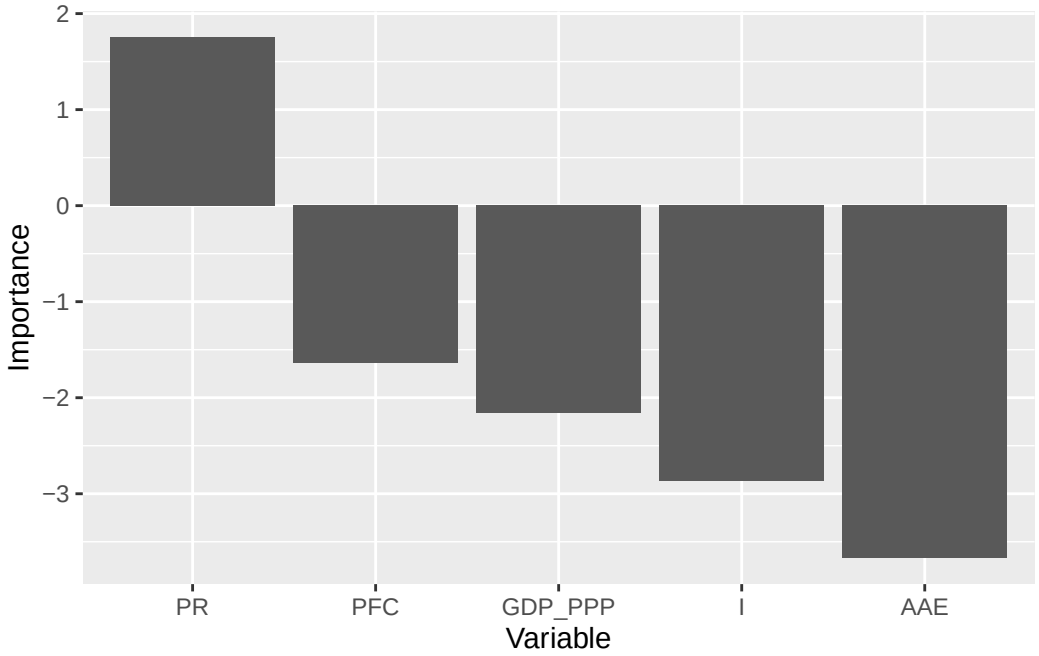


Figure 2: Barplot generated by applying 'rankingRFEAT()' to the PISAindex database to determine the ranking of variable importance.

```
efficiencyEAT(
  data, x, y, object, scores_model, digits = 3,
  FDH = TRUE, print.table = FALSE, na.rm = TRUE
)

# For Random Forest for Efficiency Analysis Trees
efficiencyRFEAT(
  data, x, y, object, digits = 3,
  FDH = TRUE, print.table = FALSE, na.rm = TRUE
)

# For Convexified Efficiency Analysis Trees
efficiencyCEAT(
  data, x, y, object, scores_model, digits = 3,
  DEA = TRUE, print.table = FALSE, na.rm = TRUE
)
```

A dataset (`data`) and the corresponding indexes of input(s) (`x`) and output(s) (`y`) must be entered. It is recommended that the `data` with the DMUs whose efficiency is to be calculated coincide with those used to estimate the frontier. However, it is also possible to calculate the efficiency scores for new `data`. The efficiency scores are calculated using the mathematical programming model included in the argument `scores_model`. The following models are available:

- `BCC.OUT`: The output-oriented radial model (Banker, Charnes, and Cooper 1984).
- `BCC.INP`: The input-oriented radial model (Banker, Charnes, and Cooper 1984).
- `RSL.OUT`: The output-oriented Russell model (Färe and Lovell 1978).
- `RSL.INP`: The input-oriented Russell model (Färe and Lovell 1978).
- `DDF`: The Directional Distance Function (Chambers, Chung, and Färe 1998).
- `WAM.MIP`: The Measure of Inefficiency Proportions as a type of Weighted Additive Model (Lovell and Pastor 1995).
- `WAM.RAM`: The Range-Adjusted Measure of Inefficiency as a type of Weighted Additive Model (Lovell and Pastor 1995; W. W. Cooper, Park, and Pastor 1999).

FDH or DEA scores can optionally be computed by setting `FDH = TRUE` or `DEA = TRUE`, respectively. Finally, a summary descriptive table of the efficiency scores can be displayed with the argument `print.table = TRUE`.

The output-oriented radial model

The output-oriented radial model determines the efficiency score for $(\mathbf{x}_k, \mathbf{y}_k) \in R_+^{m+s}$ by equiproportionally increasing all its outputs while maintaining inputs constant: $\phi(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$.

The efficiency score $\phi(\mathbf{x}_k, \mathbf{y}_k)$ can be estimated through FDH by plugging $\hat{\Psi}_{FDH}$ from (2) into $\max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$ in place of Ψ. In that case, the optimization problem can be rewritten as a mixed-integer linear optimization program, as follows:

$$\begin{aligned} \phi^{FDH}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i x_{ji} \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i y_{ri} \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i = 1, \\ & \quad \lambda_i \in \{0, 1\}, \qquad i = 1, ..., n \end{aligned} \tag{10}$$

The Linear Programming model that should be solved under Data Envelopment Analysis would be:

$$\begin{aligned} \phi^{DEA}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i x_{ji} \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i y_{ri} \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{i=1}^{n} \lambda_i = 1, \\ & \quad \lambda_i \geq 0, \qquad i = 1, ..., n \end{aligned} \tag{11}$$

The following Mixed-Integer Linear Program should be solved for Efficiency Analysis Trees:

$$\begin{aligned} \phi^{EAT}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t = 1, \\ & \quad \lambda_t \in \{0, 1\}, \qquad t \in \tilde{T}^* \end{aligned} \tag{12}$$

In R, this model can be computed by setting `scores_model = "BCC.OUT"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min Q1 Median   Q3  Max
#>    EAT 1.03      0.04   1  1   1.01 1.01 1.16
#>    FDH 1.01      0.02   1  1   1.00 1.00 1.12

scores %>% sample_n(3)

#>     EAT_BCC_OUT FDH_BCC_OUT
#> CHL        1.09        1.05
#> SAU        1.05        1.00
#> CAN        1.01        1.01
```

Finally, the optimization model that should be solved for Convexified Efficiency Analysis Trees is:

$$\begin{aligned} \phi^{CEAT}(\mathbf{x}_k, \mathbf{y}_k) = \quad & \max \quad \phi, \\ & s.t. \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk}, \qquad j = 1, ..., m \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi y_{rk}, \quad r = 1, ..., s \\ & \quad \textstyle\sum_{t \in \tilde{T}^*} \lambda_t = 1, \\ & \quad \lambda_t \geq 0, \qquad t \in \tilde{T}^* \end{aligned} \tag{13}$$

In R, this model can be computed by setting `scores_model = "BCC.OUT"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
```

```
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 1.11      0.07   1 1.05   1.09 1.09 1.31
#>    DEA 1.05      0.04   1 1.01   1.05 1.05 1.18

scores %>% sample_n(3)

#>     CEAT_BCC_OUT DEA_BCC_OUT
#> BLR         1.07        1.00
#> SVK         1.07        1.04
#> RUS         1.04        1.00
```

In the case of the output-oriented radial model, Esteve et al. (2021) showed how this measure can be computed through Random Forest where $\phi(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \Psi\}$ can be estimated by substituting the theoretical production possibility set Ψ by its estimation $\hat{\Psi}_{RF+EAT}$, i.e., $\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k) = \max\{\phi_k \in R : (\mathbf{x}_k, \phi_k \mathbf{y}_k) \in \hat{\Psi}_{RF+EAT}\}$. In particular, $\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k)$ may be calculated as:

$$\phi^{RF+EAT}(\mathbf{x}_k, \mathbf{y}_k) = \min_{r=1,\ldots,s} \left\{ \frac{y_r^{RF+EAT(\aleph)}(\mathbf{x}_k)}{y_{rk}} \right\}, \tag{14}$$

where $y_r^{RF+EAT(\aleph)}(\mathbf{x}_k)$ is the estimation of the r-th output given the input bundle $\mathbf{x}_k$.

In R, this model can be computed using `efficiencyREAT()`:

```
scores <- efficiencyRFEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelRFEAT,
                          digits = 2, print.table = TRUE)

#>  Model Mean Std. Dev.  Min Q1 Median   Q3  Max
#>  RFEAT 1.03      0.04 0.94  1   1.02 1.02 1.15
#>    FDH 1.01      0.02 1.00  1   1.00 1.00 1.12

scores %>% sample_n(3)

#>     RFEAT_BCC_OUT FDH_BCC_OUT
#> SGP          0.94           1
#> HUN          0.99           1
#> MEX          1.00           1
```

The input-oriented radial model

By analogy with the previous section, where the output-oriented radial model was shown, it is possible to calculate the input-oriented radial technical efficiency of the input-output bundle $(\mathbf{x}_k, \mathbf{y}_k)$ by solving the following Mixed-Integer Linear Program, counterpart to (10):

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{i=1}^{n} \lambda_i x_{ji} \leq \theta x_{jk}, \quad j = 1, \ldots, m \\ & \sum_{i=1}^{n} \lambda_i y_{ri} \geq y_{rk}, \quad r = 1, \ldots, s \\ & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \in \{0, 1\}, \quad i = 1, \ldots, n \end{array} \tag{15}$$

The same type of technical measure can be estimated through DEA by convexification of the production frontier generated by FDH. Next, we show the Linear Programming model that should be solved in that case:

$$\begin{array}{ll} \min & \theta, \\ s.t. & \\ & \sum_{i=1}^{n} \lambda_i x_{ji} \leq \theta x_{jk}, \quad j = 1, \ldots, m \\ & \sum_{i=1}^{n} \lambda_i y_{ri} \geq y_{rk}, \quad r = 1, \ldots, s \\ & \sum_{i=1}^{n} \lambda_i = 1, \\ & \lambda_i \geq 0, \quad i = 1, \ldots, n \end{array} \tag{16}$$

The input-oriented radial model in the case of the Efficiency Analysis Trees technique can be determined through the following Mixed-Integer Linear Program:

$$\begin{aligned} &\min \quad \theta, \\ &s.t. \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t a_j^t \leq \theta x_{jk}, && j = 1,...,m \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, && r = 1,...,s \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t = 1, \\ &\qquad \lambda_t \in \{0,1\}, && t \in \tilde{T}^* \end{aligned} \tag{17}$$

In R, this model can be computed by setting `scores_model = "BCC.INP"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "BCC.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
#>    EAT 0.94      0.06 0.69 0.90   0.96 0.96   1
#>    FDH 0.98      0.03 0.90 0.97   1.00 1.00   1

scores %>% sample_n(3)

#>     EAT_BCC_INP FDH_BCC_INP
#> DEU        0.88        0.92
#> KAZ        1.00        1.00
#> SRB        0.99        1.00
```

Additionally, under the Convexified Efficiency Analysis Trees technique, the optimization model corresponding to the convexification of the production possibility set derived from $conv(\hat{\Psi}_{T^*})$ from (5) should be solved in order to determine an estimation of the input-oriented radial measure as follows:

$$\begin{aligned} &\min \quad \theta, \\ &s.t. \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t a_j^t \leq \theta x_{jk}, && j = 1,...,m \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, && r = 1,...,s \\ &\qquad \textstyle\sum_{t\in\tilde{T}^*} \lambda_t = 1, \\ &\qquad \lambda_t \geq 0, && t \in \tilde{T}^* \end{aligned} \tag{18}$$

In R, this model can be computed by setting `scores_model = "BCC.INP"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "BCC.INP", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3 Max
#>   CEAT 0.82      0.08 0.69 0.76   0.81 0.81   1
#>    DEA 0.92      0.07 0.72 0.87   0.91 0.91   1

scores %>% sample_n(3)

#>     CEAT_BCC_INP DEA_BCC_INP
#> QAT         0.93        1.00
#> CYP         0.69        0.78
#> CHE         0.73        0.85
```

The output-oriented Russell measure

The output-oriented Russell measure under FDH must be calculated through the following optimization model:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{i=1}^{n}\lambda_t x_{ji} \leq x_{jk}, \qquad j=1,...,m \\
& \sum_{i=1}^{n}\lambda_t y_{ri} \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{i=1}^{n}\lambda_t = 1, \\
& \lambda_i \in \{0,1\}, \qquad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{19}
$$

Under DEA, the corresponding model would be:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{i=1}^{n}\lambda_t x_{ji} \leq x_{jk}, \qquad j=1,...,m \\
& \sum_{i=1}^{n}\lambda_t y_{ri} \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{i=1}^{n}\lambda_t = 1, \\
& \lambda_i \geq 0, \qquad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{20}
$$

If we resort to the Efficiency Analysis Trees technique, then the model to be solved should be the following:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk}, \qquad j=1,...,m \\
& \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}(\mathbf{a}^t) \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\
& \lambda_t \in \{0,1\}, \qquad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{21}
$$

In R, this model can be computed by setting `scores_model = "RSL.OUT"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "RSL.OUT", digits = 2,
                        print.table = TRUE)

scores %>% sample_n(3)
```

Finally, under the Convexified Efficiency Analysis Trees technique, the model would be:

$$
\begin{array}{ll}
\max & \frac{1}{s}\sum_{r=1}^{s}\phi_r, \\
s.t. & \\
& \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk}, \qquad j=1,...,m \\
& \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq \phi_r y_{rk}, \quad r=1,...,s \\
& \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\
& \lambda_t \geq 0, \qquad i=1,...,n \\
& \boldsymbol{\phi} \geq \mathbf{1}_s.
\end{array}
\tag{22}
$$

In R, this model can be computed by setting `scores_model = "RSL.OUT"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "RSL.OUT", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 1.13      0.08   1 1.07   1.10 1.10 1.34
#>    DEA 1.06      0.05   1 1.02   1.06 1.06 1.22

scores %>% sample_n(3)

#>     CEAT_RSL_OUT DEA_RSL_OUT
#> LVA         1.07        1.05
#> CHL         1.18        1.13
#> MAR         1.14        1.00
```

The input-oriented Russell measure

By analogy with the output-oriented Russell measure, the input-oriented Russell measure should be calculated through the following optimization models, depending on the selected approach:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{i=1}^{n}\lambda_t x_{ji} \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{i=1}^{n}\lambda_t y_{ri} \geq y_{rk}, \quad r=1,...,s \\ & \sum_{i=1}^{n}\lambda_t = 1, \\ & \lambda_i \in \{0,1\}, \quad i=1,...n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{23}$$

Under DEA, the corresponding model would be:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{i=1}^{n}\lambda_t x_{ji} \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{i=1}^{n}\lambda_t y_{ri} \geq y_{rk}, \quad r=1,...,s \\ & \sum_{i=1}^{n}\lambda_t = 1, \\ & \lambda_i \geq 0, \quad i=1,...n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{24}$$

If we resort to the Efficiency Analysis Trees technique, then the model to be solved should be the following:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r=1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \in \{0,1\}, \quad i=1,...,n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{25}$$

In R, this model can be computed by setting `scores_model = "RSL.INP"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "RSL.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>    EAT 0.58      0.09 0.43 0.52   0.56 0.56 0.81
#>    FDH 0.87      0.10 0.59 0.81   0.86 0.86 1.00

scores %>% sample_n(3)

#>     EAT_RSL_INP FDH_RSL_INP
#> LBN        0.58        0.73
#> MAR        0.69        0.97
#> MKD        0.58        0.87
```

Finally, under the Convexified Efficiency Analysis Trees technique, the model would be:

$$\begin{array}{ll} \min & \frac{1}{m}\sum_{j=1}^{m}\theta_j, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq \theta_j x_{jk}, \quad j=1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk}, \quad r=1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \geq 0, \quad i=1,...,n \\ & \boldsymbol{\theta} \leq \mathbf{1}_m. \end{array} \tag{26}$$

In R, this model can be computed by setting `scores_model = "RSL.INP"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
```

```
                        scores_model = "RSL.INP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>   CEAT 0.54      0.08 0.43 0.49   0.53 0.53 0.79
#>    DEA 0.80      0.11 0.59 0.74   0.78 0.78 1.00

scores %>% sample_n(3)

#>     CEAT_RSL_INP DEA_RSL_INP
#> ARG         0.44        0.59
#> LUX         0.44        0.65
#> CHE         0.49        0.74
```

The directional distance function

Chambers, Chung, and Färe (1998) introduced the directional distance function (DDF) as a technical efficiency measure that projects (x_k, y_k) through a pre-assigned direction $\mathbf{g} = (-\mathbf{g}_j^-, +\mathbf{g}_r^+) \neq 0_{m+s}, \mathbf{g}_j^- \in R^m, \mathbf{g}_r^+ \in R^s$ to the efficiency frontier of the corresponding technology. Under FDH, the DDF is calculated as follows:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{i=1}^n \lambda_i x_{ji} \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{i=1}^n \lambda_i y_{ri} \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{i=1}^n \lambda_i = 1, & \\ & \lambda_i \in \{0, 1\}, & i = 1, ..., n \end{array} \tag{27}$$

The corresponding linear program in DEA is as follows:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{i=1}^n \lambda_i x_{ji} \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{i=1}^n \lambda_i y_{ri} \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{i=1}^n \lambda_i = 1, & \\ & \lambda_i \geq 0, & i = 1, ..., n \end{array} \tag{28}$$

In the context of Efficiency Analysis Trees, the DDF is calculated through the following Mixed-Integer Linear Program:

$$\begin{array}{lll} \max & \beta_k, & \\ s.t. & & \\ & \sum_{t \in \tilde{T}^*} \lambda_t a_j^t \leq x_{jk} - \beta_k g_j^-, & j = 1, ..., m \\ & \sum_{t \in \tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + \beta_k g_r^+, & r = 1, ..., s \\ & \sum_{t \in \tilde{T}^*} \lambda_t = 1, & \\ & \lambda_t \in \{0, 1\}, & t \in \tilde{T}^*. \end{array} \tag{29}$$

In R, this model can be computed by setting `scores_model = "DDF"` in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "DDF", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev. Min Q1 Median   Q3  Max
#>    EAT 0.02      0.02   0  0   0.01 0.01 0.13
#>    FDH 0.01      0.01   0  0   0.00 0.00 0.05

scores %>% sample_n(3)

#>     EAT_DDF FDH_DDF
#> MDA    0.00    0.00
#> LVA    0.00    0.00
#> BRA    0.03    0.01
```

In the case of Convexified Efficiency Analysis Trees, the optimization model is as follows.

$$\begin{array}{ll} \max & \beta_k, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk} - \beta_k g_j^-, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + \beta_k g_r^+, \quad r = 1,...,s \\ & \sum_{t\in\tilde{T}^*}\lambda_t = 1, \\ & \lambda_t \geq 0, \qquad t \in \tilde{T}^*. \end{array} \tag{30}$$

In R, this model can be computed by setting `scores_model = "DDF"` in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "DDF", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev. Min   Q1 Median   Q3  Max
#>   CEAT 0.07      0.04   0 0.05   0.06 0.06 0.18
#>    DEA 0.03      0.03   0 0.00   0.03 0.03 0.12

scores %>% sample_n(3)

#>     CEAT_DDF DEA_DDF
#> IRL     0.05    0.03
#> LBN     0.15    0.10
#> FRA     0.07    0.06
```

The weighted additive model

The additive model measures technical efficiency based on input excesses and output shortfalls. It characterizes efficiency in terms of the input and output slacks: $\mathbf{s}^- \in R^m$ and $\mathbf{s}^+ \in R^s$, respectively. The **eat** package implements the weighted additive model formulation of Lovell and Pastor (1995), where $(\mathbf{w}^-, \mathbf{w}^+) \in R_+^m \times R_+^s$ are the input and output weights whose elements can vary across DMUs.

In the case of the FDH, the optimization program to be solved would be:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{i=1}^n \lambda_t x_{ji} \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{i=1}^n \lambda_t y_{ri} \geq y_{rk} + s_{rk}^+, \qquad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \in \{0,1\}, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{31}$$

Under DEA, the model would be as follows:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{i=1}^n \lambda_t x_{ji} \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{i=1}^n \lambda_t y_{ri} \geq y_{rk} + s_{rk}^+, \qquad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \geq 0, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{32}$$

Within the framework of Efficiency Analysis Trees, the weighted additive model would be calculated as follows:

$$\begin{array}{ll} \max & \sum_{j=1}^m w_j^- s_{jk}^- + \sum_{r=1}^s w_r^+ s_{rk}^+, \\ s.t. & \\ & \sum_{t\in\tilde{T}^*}\lambda_t a_j^t \leq x_{jk} - s_{jk}^-, \qquad j = 1,...,m \\ & \sum_{t\in\tilde{T}^*}\lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + s_{rk}^+, \quad r = 1,...,s \\ & \sum_{i=1}^n \lambda_i = 1, \\ & \lambda_i \in \{0,1\}, \qquad i = 1,...,n \\ & \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. \end{array} \tag{33}$$

In R, this model can be computed by setting `scores_model = "WAM.MIP"` for the Measure of Inefficiency Proportions or `"WAM.RAM"` for the Range-Adjusted Measure of Inefficiency (W. W. Cooper, Park, and Pastor 1999) in `efficiencyEAT()`:

```
scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "WAM.MIP", digits = 2,
                        print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>    EAT 2.07      0.56 0.61 1.58   2.18 2.18 3.14
#>    FDH 0.40      0.58 0.00 0.00   0.00 0.00 2.37

scores %>% sample_n(3)

#>     EAT_WAM_MIP FDH_WAM_MIP
#> MLT        2.54           0
#> SVN        1.86           0
#> HRV        2.30           0

scores <- efficiencyEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                        scores_model = "WAM.RAM", digits = 2,
                        print.table = TRUE)

#>  Model    Mean Std. Dev.   Min      Q1  Median      Q3     Max
#>    EAT 2704.65   1677.11 455.3 1467.78 2425.09 2425.09 8293.59
#>    FDH  959.09   1388.56   0.0    0.00    0.00    0.00 5413.27

scores %>% sample_n(3)

#>     EAT_WAM_RAM FDH_WAM_RAM
#> ITA     2192.98        0.00
#> LVA     1890.38        0.00
#> CAN     1724.26     1182.66
```

And, finally, the Convexified Efficiency Analysis Trees weighted additive model would be:

$$
\begin{aligned}
\max \quad & \textstyle\sum_{j=1}^{m} w_j^- s_{jk}^- + \sum_{r=1}^{s} w_r^+ s_{rk}^+, \\
s.t. \quad & \\
& \textstyle\sum_{t\in\tilde{T}^*} \lambda_t a_j^t \leq x_{jk} - s_{jk}^-, && j = 1,...,m \\
& \textstyle\sum_{t\in\tilde{T}^*} \lambda_t d_{rT^*}\left(\mathbf{a}^t\right) \geq y_{rk} + s_{rk}^+, && r = 1,...,s \\
& \textstyle\sum_{i=1}^{n} \lambda_i = 1, && \\
& \lambda_i \geq 0, && i = 1,...,n \\
& \mathbf{s}_k^- \geq \mathbf{0}_m, \mathbf{s}_k^+ \geq \mathbf{0}_s. &&
\end{aligned}
\tag{34}
$$

In R, this model can be computed by setting `scores_model = "WAM.MIP"` for the Measure of Inefficiency Proportions or `"WAM.RAM"` for the Range-Adjusted Measure of Inefficiency in `efficiencyCEAT()`:

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "WAM.MIP", digits = 2,
                         print.table = TRUE)

#>  Model Mean Std. Dev.  Min   Q1 Median   Q3  Max
#>   CEAT 2.39      0.45 0.96 2.24   2.50 2.50 3.14
#>    DEA 0.90      0.62 0.00 0.23   1.07 1.07 2.37

scores %>% sample_n(3)

#>     CEAT_WAM_MIP DEA_WAM_MIP
#> ISR         2.74        1.13
#> ITA         2.73        1.31
#> SGP         1.91        0.00
```

```
scores <- efficiencyCEAT(data = PISAindex, x = 4:8, y = 1:3, object = modelEAT,
                         scores_model = "WAM.RAM", digits = 2,
                         print.table = TRUE)

#>  Model    Mean Std. Dev.    Min      Q1  Median      Q3      Max
#>   CEAT 5285.99   2573.13 612.54 3362.98 4836.79 4836.79 12088.64
#>    DEA 2413.66   1821.33   0.00  746.22 2495.20 2495.20  7167.29

scores %>% sample_n(3)

#>     CEAT_WAM_RAM DEA_WAM_RAM
#> KOR      1538.61        0.00
#> ROU      7239.16     4727.17
#> EST       612.54        0.00
```

6 Advanced options and displaying and exporting results

Advanced optimization options

The `bestEAT()` and `bestRFEAT()` functions are aimed at finding the value of the hyperparameters that minimize the root mean squared error (RMSE) calculated from a test sample through an Efficiency Analysis Trees or a Random Forest for Efficiency Analysis Trees model fitted using a training sample. The code of these functions is as follows:

```
# Hyperparameter tuning for Efficiency Analysis Trees
bestEAT(
  training, test, x, y,
  numStop = 5, fold = 5,
  max.depth = NULL,
  max.leaves = NULL,
  na.rm = TRUE
  )

# Hyperparameter tuning for Random Forest for Efficiency Analysis Trees
bestRFEAT(
  training, test, x, y,
  numStop = 5, m = 50,
  s_mtry = c("5", "BRM"),
  na.rm = TRUE
  )
```

Here is an example of using the `bestEAT()` function. First, the PISAindex database explained in Section Data structure is divided into a training subset with 70% of the DMUs and a test subset with the remaining 30% (these values can be modified).

```
n <- nrow(PISAindex)              # Observations in the dataset
selected <- sample(1:n, n * 0.7)  # Training indexes
training <- PISAindex[selected, ] # Training set
test <- PISAindex[- selected, ]   # Test set
```

Then, we can apply the `bestEAT()` function. This function, and its equivalent `bestRFEAT()`, requires a training set (`training`) on which to fit an Efficiency Analysis Trees model (with cross-validation), a test set (`test`) on which to calculate the root mean squared error and the input and output indexes (`x` and `y`, respectively). The rest of the arguments (`numStop`, `fold`, `max.depth` and `max.leaves` in case of using the `bestEAT()` function) are used to create a grid of combinations that determines the number of models to fit. Notice that it is not possible to enter `NULL` and a certain value in `max.depth` or `max.leaves` arguments at the same time (i.e. `max.depth = c(NULL, 5, 3)`).

In the following example, the arguments `numStop = (3, 5, 7)` and `fold = (5, 7)` are entered and, consequently, six different models are constructed and fitted with {`numStop = 3`, `fold = 5`}, {`numStop = 3`, `fold = 7`}, {`numStop = 5`, `fold = 5`}, {`numStop = 5`, `fold = 7`}, {`numStop = 7`, `fold = 5`} and {`numStop = 7`, `fold = 7`}. Let us show a numerical example:

```
bestEAT(training = training, test = test, x = 4:8, y = 1:3,
        numStop = c(3, 5, 7), fold = c(5, 7))

#>   numStop fold   RMSE leaves
#> 1       5    5  74.74     15
#> 2       7    7  83.61     13
#> 3       3    5  84.17     13
#> 4       3    7  84.17     13
#> 5       7    5  86.39      8
#> 6       5    7 104.93      8
```

The best model is given by the hyperparameters {`numStop = 5`, `fold = 5`} with RMSE = 74.74 and 15 leaf nodes. Note that sometimes it might be interesting to select a model with a higher RMSE but with a lower number of leaf nodes. With this result, we fit the final Efficiency Analysis Trees model using all the original data.

```
bestEAT_model <- EAT(data = PISAindex, x = 4:8, y = 1:3, numStop = 5, fold = 5)
```

Displaying results

General functions for the EAT object

The simplest functions to use in order to explore the results of an `EAT` object are `print()` and `summary()`. The function `print()` returns the tree-structure of an Efficiency Analysis Trees model; while the function `summary()` returns general information about the fitted model. We show the results with an example:

```
modelEAT2 <- EAT(data = PISAindex, x = 7, y = 3)

print(modelEAT2) # [node] y: [prediction] || R: error n(t): nº of DMUs

#>  [1] y: [ 569 ] || R: 15724.19 n(t): 72
#>
#>  |  [2] AAE < 70.12 --> y: [ 486 ] || R: 3094.43 n(t): 34
#>
#>  |   |  [4] AAE < 60.75 --> y: [ 472 ] <*> || R: 1553.93 n(t): 17
#>
#>  |   |  [5] AAE >= 60.75 --> y: [ 486 ] <*> || R: 1006.94 n(t): 17
#>
#>  |  [3] AAE >= 70.12 --> y: [ 569 ] <*> || R: 3753.38 n(t): 38
#>
#> <*> is a leaf node

# Primary & surrogate splits: Node i --> {SL, SR} || var --> {R: error, s: threshold}
summary(modelEAT2)

#>
#>   Formula:  M_PISA ~ AAE
#>
#>  # ========================== #
#>  #   Summary for leaf nodes   #
#>  # ========================== #
#>
#>  id n(t)  % M_PISA    R(t)
#>   3   38 53    569 3753.38
#>   4   17 24    472 1553.93
#>   5   17 24    486 1006.94
#>
#>  # ========================== #
#>  #            Tree            #
#>  # ========================== #
#>
#>  Interior nodes: 2
```

```
#>      Leaf nodes: 3
#>     Total nodes: 5
#>
#>            R(T): 6314.25
#>         numStop: 5
#>            fold: 5
#>       max.depth:
#>      max.leaves:
#>
#>  # ========================== #
#>  # Primary & surrogate splits #
#>  # ========================== #
#>
#>  Node 1 --> {2,3} || AAE --> {R: 6847.81, s: 70.12}
#>
#>  Node 2 --> {4,5} || AAE --> {R: 2560.88, s: 60.75}
```

Representing the efficiency scores

`efficiencyJitter()` returns a jitter plot from `ggplot2`. This graphic shows how DMUs are grouped into leaf nodes in a model built using the `EAT()` function where each leaf node groups DMUs with the same level of resources. A black dot and a black line represent, respectively, the mean value and the standard deviation of the scores (`df_scores` from the `efficiencyEAT()` or the `efficiencyCEAT()` functions) of a given node. Additionally, efficient DMU labels are always displayed based on the model entered in the `scores_model` argument. Finally, the user can specify an upper bound (`upb`) and a lower bound (`lwb`) in order to show, in addition, the labels whose efficiency score lies between them. The code is as follows:

```
efficiencyJitter(
  object,
  df_scores,
  scores_model,
  upb = NULL,
  lwb = NULL
)
```

As an example, using data from Section Data structure, we create a new Efficiency Analysis Trees model containing only the `AAE` and the `M_PISA` variables. Next, we evaluate the Efficiency Analysis Trees efficiency scores corresponding to the output-oriented radial model and plot them through `efficiencyJitter()`.

```
scores <- efficiencyEAT(data = PISAindex, x = 7, y = 3, object = modelEAT2,
                        scores_model = "BCC.OUT", digits = 2,
                        print.table = FALSE)
```

`efficiencyDensity()` returns a density plot from `ggplot2`. This graphic allows to verify the similarity between the scores obtained by the different available methodologies (`EAT`, `FDH`, `CEAT`, `DEA` and `RFEAT`) in the **eat** package.

```
efficiencyDensity(
  df_scores,
  model = c("EAT", "FDH")
)
```

In this case, a comparison between the scores of the EAT and FDH models is shown, where it can be clearly seen how FDH is less restrictive when determining a unit as efficient:

Other graphics

In the limited case of using only one input for producing only one output, we can display the frontier (from `ggplot2`) estimated by the `EAT()` function through the `frontier()` function:

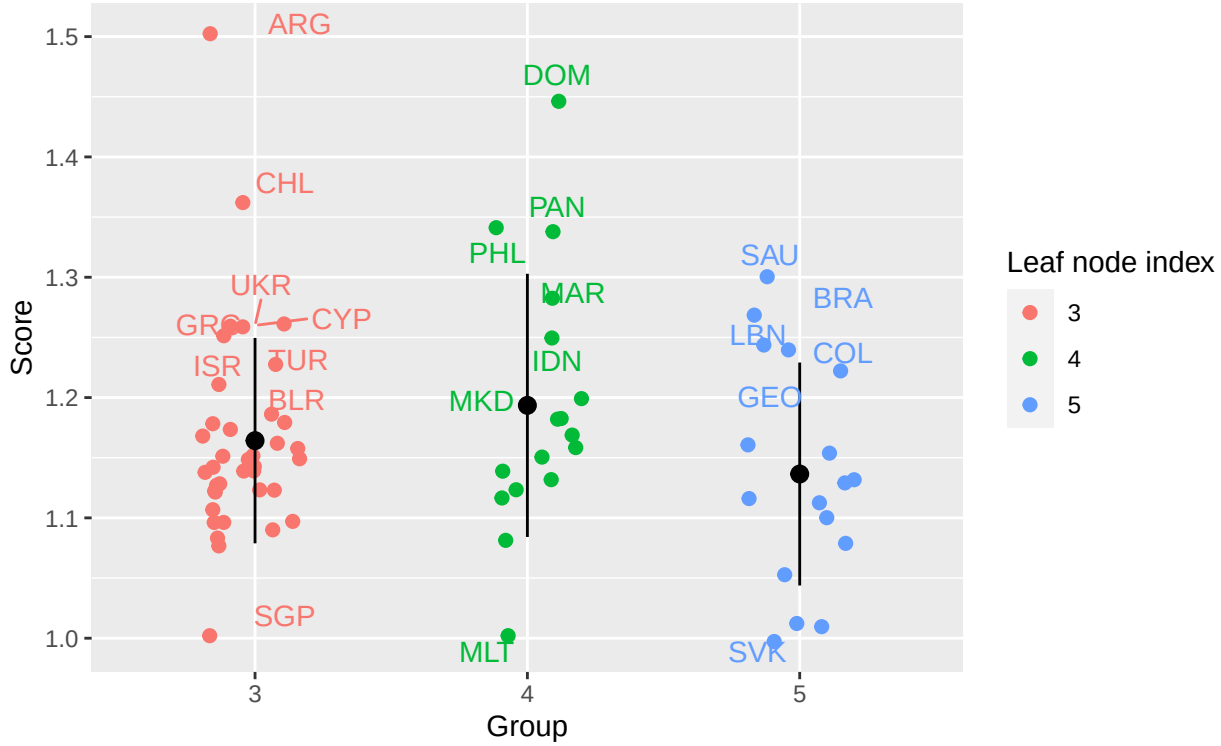


Figure 3: Jitter plot generated by 'efficiencyJitter()' to show how the countries are grouped inside three particular leaf nodes.

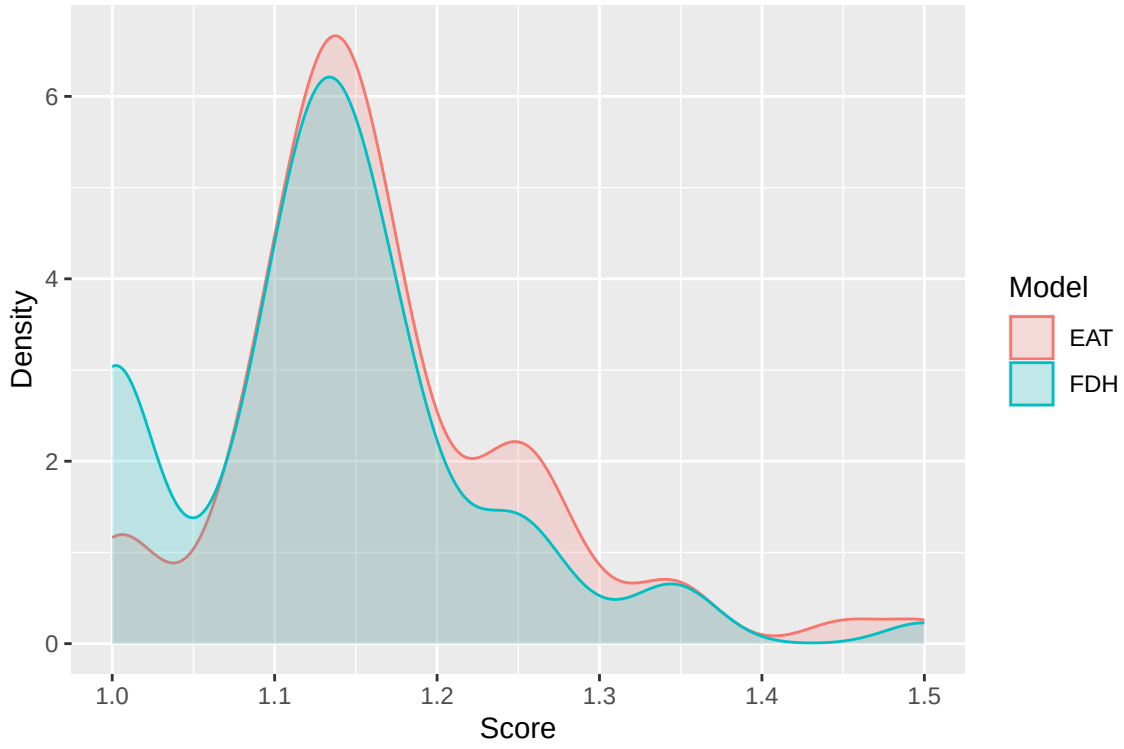


Figure 4: Density plot generated by 'efficiencyDensity()' to show the difference between the score obtained by EAT and FDH.

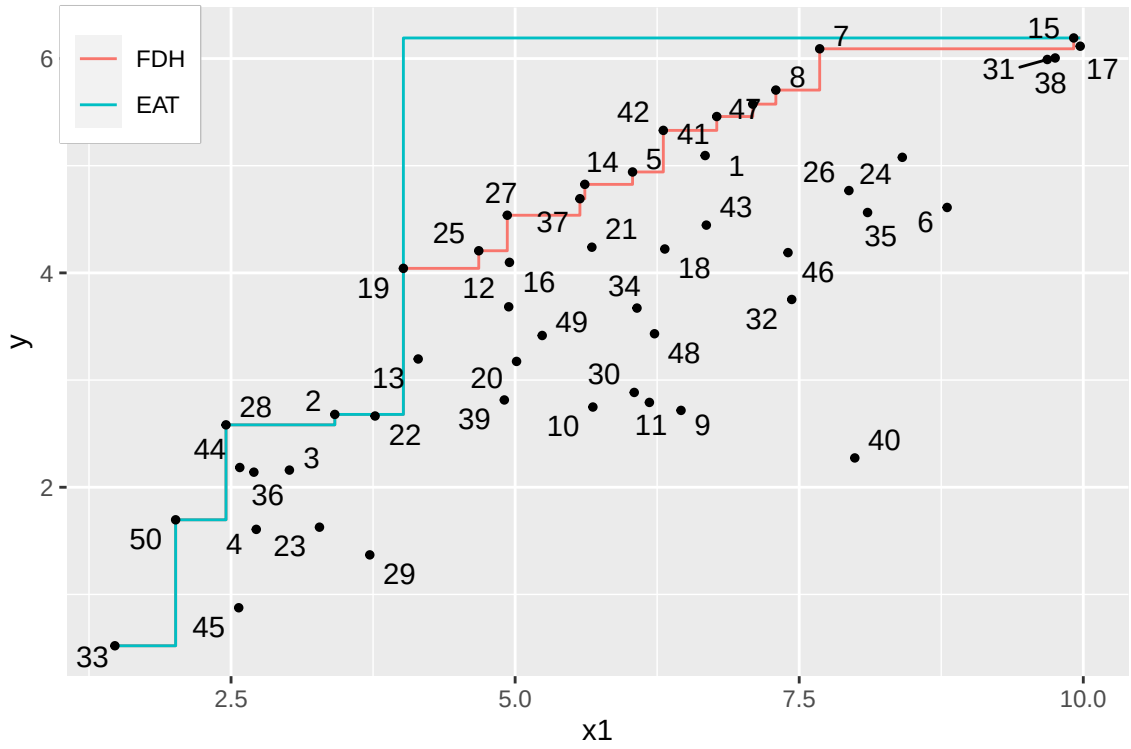


Figure 5: Plot of productions functions corresponding to the EAT and the FDH estimator when 'frontier()' is applied.

```
frontier(
  object, FDH = TRUE,
  observed.data = TRUE,
  observed.color = "black",
  pch = 19,vsize = 1,
  rwn = FALSE,
  max.overlaps = 10
)
```

Optionally, the frontier estimated by FDH can also be plotted if `FDH = TRUE`. Observed DMUs can be showed by a scatterplot if `observed.data = TRUE` and its color, shape and size can be modified with `observed.color`, `pch` and `size` respectively. Finally, row names can be included with `rwn = TRUE`.

As an example, we use data simulated from the **eat** package to generate a `data.frame` with 50 rows (`N` = DMUs) and 1 input (`nX`):

```
simulated <- Y1.sim(N = 50, nX = 1)
modelEAT3 <- EAT(data = simulated, x = 1, y = 2)
```

Then, we apply the `frontier()` function, where it can be observed how the Efficiency Analysis Trees model generalizes the results obtained by the FDH model:

The function `frontier()` shown above only works for the simple case of a low-dimensional scenario with one input and one output. For multiple input and/or output scenarios, the typical tree-structure showing the relationships between outputs and inputs is given by the function `plotEAT()`.

```
plotEAT(
  object
)
```

The nodes of the tree are colored according to the variable by which the split is performed or they are black, in the case of being a leaf node. For each node, we can obtain the following information:

- `id`: node index.

- `R`: error at the node.
- `n(t)`: number of DMUs at the node.
- input variable associated with the split.
- `y`: vector of output predictions.

Next, we limit the growth of an Efficiency Analysis Trees model to a maximum size of 5 (`max.depth = 4`) and display the tree-structure using the `plotEAT()` function:

Finally, the function `plotRFEAT()` returns the Out-Of-Bag error for a random forest consisting of k trees. The code of the function and an example with the object `modelRFEAT` are shown above:

```
plotRFEAT(
  object
)
```

In view of the results, it can be seen how the OOB error presents a great variability for a small number of trees, however, it usually levels off. In our case, it seems that the OOB error levels off from 20 trees onwards around an OOB error of 56, so it could be interesting not to include a greater number of trees in the random forest in order to reduce the computational cost.

7 Conclusions

The **eat** package allows the estimation of production frontiers in microeconomics and engineering through suitable adaptations of Regression Trees and Random Forest. In the first case, the package implements in R the so-called Efficiency Analysis Trees (EAT) by Esteve et al. (2020), which is a non-parametric technique that competes against the more standard Free Disposal Hull (FDH) technique. In this regard, the EAT technique overcomes the overfitting problem suffered by the FDH technique. FDH is based on three microeconomic postulates. First, the technology determined by FDH satisfies free disposability in inputs and outputs. Second, it is assumed to be deterministic, that is, the production possibility set built by this technique always contains all the observations that belong to the data sample. Third, FDH meets the minimal extrapolation principle. This last postulate implies that FDH generates the smallest set that satisfies the first two postulates. Consequently, the derived efficient frontier is as close to the data as possible, generating overfitting problems. In contrast, the Efficiency Analysis Trees (EAT) technique meets the first two postulates but does not satisfy the minimal extrapolation principle. This fact avoids possible overfitting problems. The difficulty for non-overfitted models lies in where to locate the production possibility set in such a way that it is close to the (unknown) technology associated with the underlying Data Generating Process. In the case of EAT, it is achieved through cross-validation and pruning. A subsequent convexification of the EAT estimation of the technology, known as CEAT by its acronym, yields an alternative estimate of the production possibility set in contrast to the traditional Data Envelopment Analysis (DEA) technique. In the second case, an ensemble of tree models is fitted and aggregated with the objective of achieving robustness in the estimation of the production frontier (Esteve et al. 2021).

Several functions have been implemented in the **eat** package for determining the best model, through a pruning process based on cross-validation, graphing the results, calculating a ranking of importance of inputs and comparing the efficiency scores estimated by EAT with respect to the standard approaches, i.e., FDH and DEA, through a list of standard technical efficiency measures. We refer to the input and output-oriented radial models, the input and output-oriented Russell measures, the Directional Distance Function and the Weighted Additive model.

Throughout the paper, we have also shown how to organize the data, use the available functions, and interpret the results. In particular, to illustrate the different functions implemented in the package, we applied all of them on a common empirical example so that results can easily be compared. In this way, we believe that the **eat** package is a valid self-contained R package for the measurement of technical efficiency from the popular machine learning technique: Decision Trees. Finally, since the code is freely available in an open source repository, users will benefit from the collaboration and review of the community. Users may check and modify the code to adapt it to their own needs and extend it with new definitions.

8 Acknowledgments

M. Esteve, V. España and J. Aparicio thank the grant PID2019-105952GB-I00 funded by Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación /10.13039/501100011033. Additionally, M.

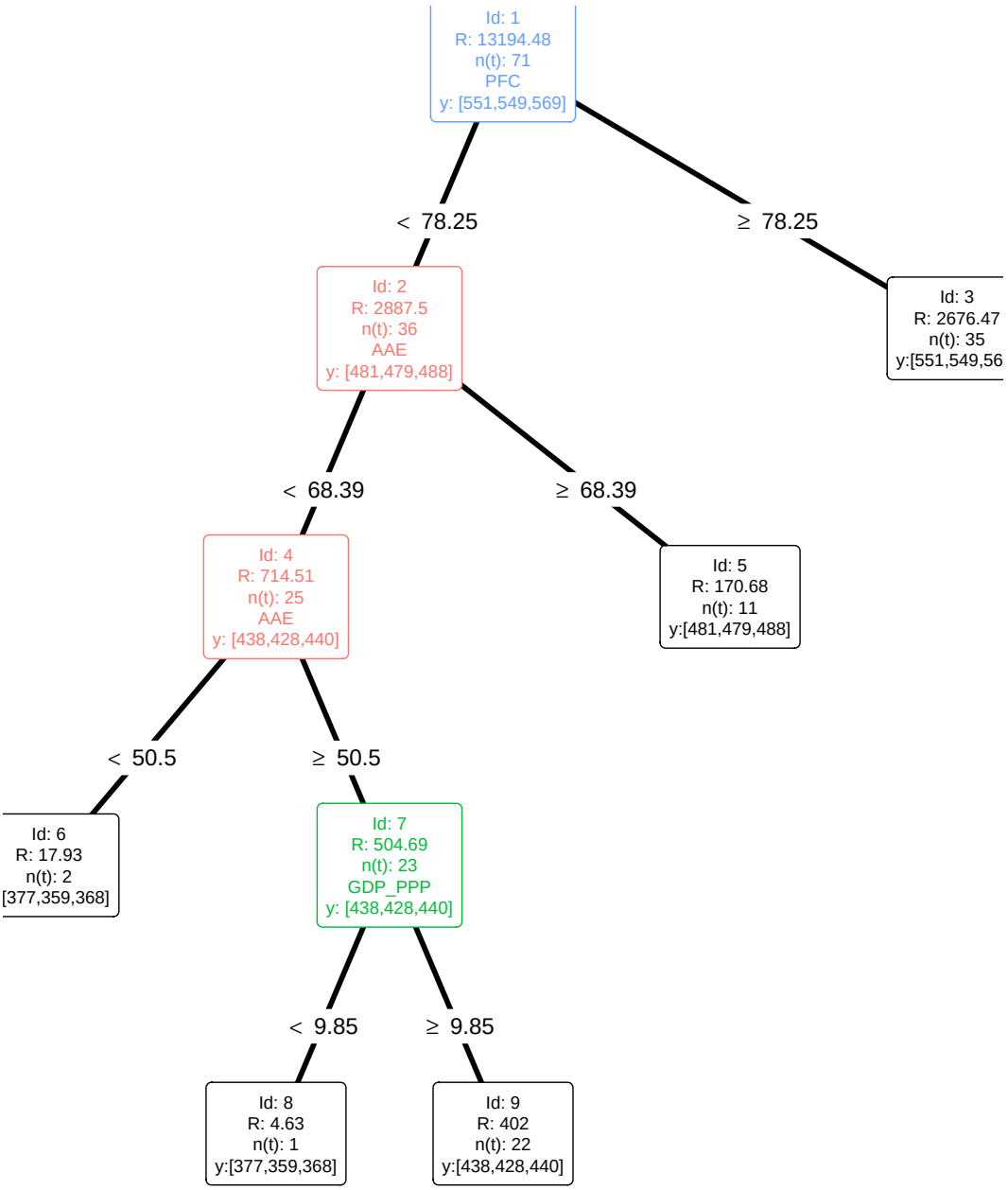


Figure 6: Plot of the tree structure obtained through an EAT model with the parameter max.depth defined as 4.

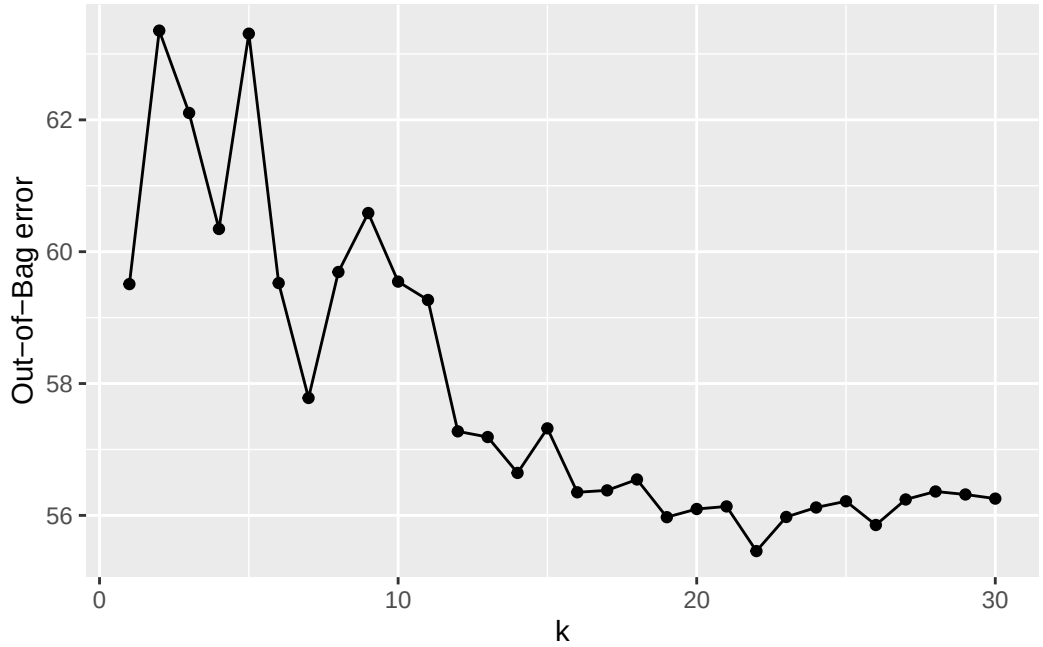


Figure 7: Plot of the OOB error corresponding to 30 different RFEAT where k represents the number of trees belonging to each RF.

Esteve gratefully acknowledges the financial support from the Spanish Ministry of Science, Innovation and Universities under Grant FPU17/05365. X. Barber gratefully acknowledges the financial support from the Spanish Ministry of Science and the State Research Agency under grant PID2019-106341GB-I00. X Barber and J. Aparicio gratefully acknowledge the financial support from the University Miguel Hernandez and the Vice-Rectorate for Research under grant AW1020IP-2020/NAC/00073. This work was also supported by the Generalitat Valenciana under Grant ACIF/2021 (V. España).

References

Afriat, Sidney N. 1972. "Efficiency Estimation of Production Functions." *International Economic Review*, 568–98.

Álvarez, Inmaculada, Javier Barbero, and José Zofio. 2020. "A Data Envelopment Analysis Toolbox for MATLAB." *Journal of Statistical Software (Online)* 95 (3).

Aparicio, Juan, Miriam Esteve, Jesus J Rodriguez-Sala, and José Zofio. 2021. "The Estimation of Productive Efficiency Through Machine Learning Techniques: Efficiency Analysis Trees." In *Data-Enabled Analytics: DEA for Big Data*, edited by Joe Zhu and Vicent Charles. Springer.

Aparicio, Juan, Jesús T Pastor, Fernando Vidal, and José L Zof 'io. 2017. "Evaluating Productive Performance: A New Approach Based on the Product-Mix Problem Consistent with Data Envelopment Analysis." *Omega* 67: 134–44.

Arnaboldi, Michela, Giovanni Azzone, and Marco Giorgino. 2014. *Performance Measurement and Management for Engineers*. Academic Press.

Balk, Bert M, Javier Barbero, and José L Zof 'Io. 2018. "A Total Factor Productivity Toolbox for MATLAB." *Available at SSRN 3178911*.

Banker, Rajiv D. 1993. "Maximum Likelihood, Consistency and Data Envelopment Analysis: A Statistical Foundation." *Management Science* 39 (10): 1265–73.

———. 1996. "Hypothesis Tests Using Data Envelopment Analysis." *Journal of Productivity Analysis* 7 (2): 139–59.

Banker, Rajiv D, Abraham Charnes, and William Wager Cooper. 1984. "Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis." *Management Science* 30 (9): 1078–92.

Banker, Rajiv D, Abraham Charnes, William W Cooper, and Roger Clarke. 1989. "Constrained Game Formulations and Interpretations for Data Envelopment Analysis." *European Journal of Operational Research* 40 (3): 299–308.

Banker, Rajiv D, and Ajay Maindiratta. 1992. "Maximum Likelihood Estimation of Monotone and Concave Production Frontiers." *Journal of Productivity Analysis* 3 (4): 401–15.

Berk, Richard A. 2016. *Statistical Learning from a Regression Perspective*. Vol. 14. Springer.
Bogetoft, Peter, and Lars Otto. 2010. *Benchmarking with DEA, SFA, and r*. Vol. 157. Springer Science & Business Media.
Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.
Breiman, Leo, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and Regression Trees*. CRC press.
Chambers, Robert G, Yangho Chung, and Rolf Färe. 1998. "Profit, Directional Distance Functions, and Nerlovian Efficiency." *Journal of Optimization Theory and Applications* 98 (2): 351–64.
Charles, Vincent, Juan Aparicio, and Joe Zhu. 2020. *Data Science and Productivity Analytics*. Springer.
Charnes, Abraham, William W Cooper, and Edwardo Rhodes. 1978. "Measuring the Efficiency of Decision Making Units." *European Journal of Operational Research* 2 (6): 429–44.
Cooper, William W, Kyung Sam Park, and Jesus T Pastor. 1999. "RAM: A Range Adjusted Measure of Inefficiency for Use with Additive Models, and Relations to Other Models and Measures in DEA." *Journal of Productivity Analysis* 11 (1): 5–42.
Cooper, WW, LM Seiford, K Tone, and J Zhu. 2007. "Some Models and Measures for Evaluating Performances with DEA: Past Accomplishments and Future Prospects." *Journal of Productivity Analysis* 28 (3): 151–63.
Daouia, Abdelaati, Thibault Laurent, and Hohsuk Noh. 2017. "npbr: A Package for Nonparametric Boundary Regression in R." *Journal of Statistical Software* 79 (9): 1–43. https://doi.org/10.18637/jss.v079.i09.
Daraio, Cinzia, and Léopold Simar. 2005. "Introducing Environmental Variables in Nonparametric Frontier Models: A Probabilistic Approach." *Journal of Productivity Analysis* 24 (1): 93–121.
Debreu, Gerard. 1951. "The Coefficient of Resource Utilization." *Econometrica: Journal of the Econometric Society*, 273–92.
Deprins, D, and L Simar. 1984. "Measuring Labor Efficiency in Post Offices, the Performance of Public Enterprises: Concepts and Measurements, M. Marchand, P. Pestieau and H. Tulkens." Amsterdam, North (Holland, 243 (267.
Dyson, Robert G, Rachel Allen, Ana S Camanho, Victor V Podinovski, Claudia S Sarrico, and Estelle A Shale. 2001. "Pitfalls and Protocols in DEA." *European Journal of Operational Research* 132 (2): 245–59.
Efron, Brad. 1979. "Bootstrap Methods: Another Look at the Jackknife." *Annals of Statistics* 7 (1): 1–26.
Esteve, Miriam, Juan Aparicio, Alejandro Rabasa, and Jesus J Rodriguez-Sala. 2020. "Efficiency Analysis Trees: A New Methodology for Estimating Production Frontiers Through Decision Trees." *Expert Systems with Applications* 162: 113783.
Esteve, Miriam, Juan Aparicio, Jesus J Rodriguez-Sala, and Joe Zhu. 2021. "Estimation of Production Frontiers Through Random Forest: The Treatment of Lack of Robustness, Ranking of Inputs and Curse of Dimensionality Under Free Disposal Hull. Working Paper." Working paper.
Färe, Rolf, and CA Knox Lovell. 1978. "Measuring the Technical Efficiency of Production." *Journal of Economic Theory* 19 (1): 150–62.
Färe, Rolf, and Daniel Primont. 1995. *Multi-Output Production and Duality: Theory and Applications*. Springer Science & Business Media.
Farrel, MJ. 1957. "The Measure of Productive Efficiency." *Journal of the Royal Statistical Society* 120.
Ferrara, Giancarlo, and Francesco Vidoli. 2018. *Semsfa: Semiparametric Estimation of Stochastic Frontier Models*. https://CRAN.R-project.org/package=semsfa.
Friedman, Leo, and Zilla Sinuany Stern. 1998. "Combining Ranking Scales and Selecting Variables in the DEA Context: The Case of Industrial Branches." *Computers & Operations Research* 25 (9): 781–91.
Golany, Boaz, and Yaakov Roll. 1989. "An Application Procedure for DEA." *Omega* 17 (3): 237–50.
Homburg, Carsten. 2001. "Using Data Envelopment Analysis to Benchmark Activities." *International Journal of Production Economics* 73 (1): 51–58.
Ji, Yong-bae, and Choonjoo Lee. 2010. "Data Envelopment Analysis." *The Stata Journal* 10 (2): 267–80.
Khezrimotlagh, Dariush, Joe Zhu, Wade D Cook, and Mehdi Toloo. 2019. "Data Envelopment Analysis and Big Data." *European Journal of Operational Research* 274 (3): 1047–54.
Koopmans, Tjalling C. 1951. "An Analysis of Production as an Efficient Combination of Activities." *Activity Analysis of Production and Allocation*.
Kuhn, Max, Kjell Johnson, et al. 2013. *Applied Predictive Modeling*. Vol. 26. Springer.
Kuosmanen, Timo, and Andrew Johnson. 2017. "Modeling Joint Production of Multiple Outputs in StoNED: Directional Distance Function Approach." *European Journal of Operational Research* 262 (2): 792–801.
Kuosmanen, Timo, and Andrew L Johnson. 2010. "Data Envelopment Analysis as Nonparametric Least-Squares Regression." *Operations Research* 58 (1): 149–60.
Landete, Mercedes, Juan F Monge, and José L Ruiz. 2017. "Robust DEA Efficiency Scores: A Probabilistic/Combinatorial Approach." *Expert Systems with Applications* 86: 145–54.
Lovell, CA Knox, and Jesús T Pastor. 1995. "Units Invariant and Translation Invariant DEA Models."

Operations Research Letters 18 (3): 147–51.

McKenzie, Taylor. 2018. *Snfa: Smooth Non-Parametric Frontier Analysis*. https://CRAN.R-project.org/package=snfa.

Nunamaker, Thomas R. 1985. "Using Data Envelopment Analysis to Measure the Efficiency of Non-Profit Organizations: A Critical Evaluation." *Managerial and Decision Economics* 6 (1): 50–58.

O'Donnell, Christopher J et al. 2018. *Productivity and Efficiency Analysis*. Springer.

OECD, PISA. 2018. "Assessment and Analytical Framework, PISA." OECD Publishing. Paris., PISA (OECD, 2019).

Oh, Dong-hyun, and Dukrok Suh. 2013. *Nonparaeff: Nonparametric Methods for Measuring Efficiency and Productivity*. https://CRAN.R-project.org/package=nonparaeff.

Orea, Luis, and José Luis Zof 'io. 2019. "Common Methodological Choices in Nonparametric and Parametric Analyses of Firms' Performance." In *The Palgrave Handbook of Economic Performance Analysis*, 419–84. Springer.

Pastor, Jesús T, JosÉ L Ruiz, and Inmaculada Sirvent. 2002. "A Statistical Test for Nested Radial DEA Models." *Operations Research* 50 (4): 728–35.

Raab, Raymond L, and Richard W Lichty. 2002. "Identifying Subareas That Comprise a Greater Metropolitan Area: The Criterion of County Relative Efficiency." *Journal of Regional Science* 42 (3): 579–94.

Shephard, Ronald William. 1953. *Theory of Cost and Production Functions*. Princeton University Press.

Simar, Leopold, and Paul W Wilson. 1998. "Sensitivity Analysis of Efficiency Scores: How to Bootstrap in Nonparametric Frontier Models." *Management Science* 44 (1): 49–61.

———. 2000. "Statistical Inference in Nonparametric Frontier Models: The State of the Art." *Journal of Productivity Analysis* 13 (1): 49–78.

Simar, Léopold, and Paul W Wilson. 2000. "A General Methodology for Bootstrapping in Non-Parametric Frontier Models." *Journal of Applied Statistics* 27 (6): 779–802.

Social Progress Index. 2018. "Social Progress Index 2018." https://www.socialprogress.org/.

StataCorp. 2021. "Stata Statistical Software: Release 14." http://www.stata.com/.

The MathWorks Inc. 2021. "MATLAB – the Language of Technical Computing." http://www.mathworks.com/products/matlab/.

Zhu, Joe et al. 2019. "DEA Under Big Data: Data Enabled Analytics and Network Data Envelopment Analysis." *Annals of Operations Research*, 1–23.

Miriam Esteve
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-5908-0581
miriam.estevec@umh.es

Victor España
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-1807-6180
vespana@umh.es

Juan Aparicio
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0002-0867-0004
j.aparicio@umh.es

Xavier Barber
Miguel Hernandez University
Center of Operations Research
03202 Elche, Spain
https://cio.umh.es/
ORCiD: 0000-0003-3079-5855
xbarber@umh.es

Will the Real Hopkins Statistic Please Stand Up?

by Kevin Wright

Abstract Hopkins statistic (Hopkins and Skellam 1954) can be used to test for spatial randomness of data and for detecting clusters in data. Although the method is nearly 70 years old, there is persistent confusion regarding the definition and calculation of the statistic. We investigate the confusion and its possible origin. Using the most general definition of Hopkins statistic, we perform a small simulation to verify its distributional properties, provide a visualization of how the statistic is calculated, and provide a fast R function to correctly calculate the statistic. Finally, we propose a protocol of five questions to guide the use of Hopkins statistic.

1 Introduction

Hopkins and Skellam (1954) introduced a statistic to test for spatial randomness of data. If the null hypothesis of spatial randomness is rejected, then one possible interpretation is that the data may be clustered into distinct groups. Since one of the problems with clustering methods is that they will always identify clusters, (even if there are no meaningful clusters in the data), Hopkins statistic can be used to determine if there are clusters in the data *before* applying clustering methods. In the description below on how to calculate Hopkins statistic, we follow the terminology of earlier authors and refer to an "event" as one of the existing data values in a matrix X, and a "point" as a new, randomly chosen location. For clarity in the discussions below we make a distinction between D, the dimension of the data, and d, the exponent in the formula for Hopkins statistic.

Let X be a matrix of n events (in rows) and D variables (in columns). Let U be the space defined by X.

Hopkins statistic is calculated with the following algorithm:

1. Sample at random one of the existing events from the data X. Let w_i be the Euclidean distance from this event to the nearest-neighbor event in X.
2. Generate one new point uniformly distributed in U. Let u_i be the Euclidean distance from this point to the nearest-neighbor event in X.
3. Repeat steps (1) and (2) m times, where m is a small fraction of n, such as 10%.
4. Calculate $H = \sum_{i=1}^{m} u_i^d / \sum_{i=1}^{m} (u_i^d + w_i^d)$, where $d = D$.

Because of sampling variability, it is common to calculate H multiple times and take the average. Under the null hypothesis of spatial randomness, this statistic has a Beta(m,m) distribution and will always lie between 0 and 1. The interpretation of H follows these guidelines:

- Low values of H indicate repulsion of the events in X away from each other.
- Values of H near 0.5 indicate spatial randomness of the events in X.
- High values of H indicate possible clustering of the events in X. Values of $H > 0.75$ indicate a clustering tendency at the 90% confidence level (Lawson and Jurs 1990).

2 A short history of Hopkins statistic

There exists considerable confusion about the definition of Hopkins statistic in scientific publications. In particular, when calculating Hopkins statistic, there are 3 different values of the exponent d (in step 4 above) that have been used in statistical literature: $d = 1$, $d = 2$, and the generalized $d = D$. Here is a brief timeline of how this exponent has been presented.

- 1954: Hopkins and Skellam (1954) introduced Hopkins statistic in a two-dimensional setting. The formula they present is in a slightly different form, but is equivalent to $\sum u_i^2 / \sum (u_i^2 + w_i^2)$. The exponent here is $d = 2$.
- 1976: Diggle, Besag, and Gleaves (1976) presented a formula for Hopkins statistic in a two-dimensional setting as $\sum u_i / \sum (u_i + w_i)$. This formula has no exponents and therefore at first glance *appears* to use the exponent $d = 1$ in the equation for Hopkins statistic. However, a careful reading of their text shows that their u_i and w_i values were actually *squared Euclidean distances*. If their u_i and w_i had represented ordinary (non-squared) Euclidean distances, then their formula would have been $\sum u_i^2 / \sum (u_i^2 + w_i^2)$. We suspect this paper is the likely source of confusion by later authors.

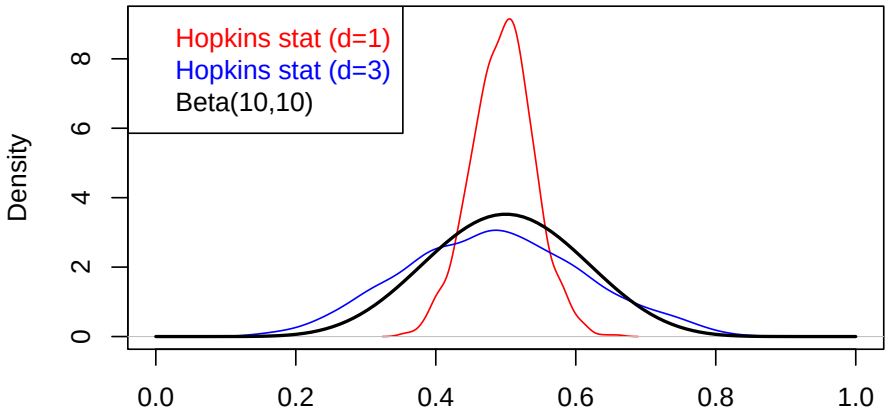


Figure 1: Results of a simulation study of the distribution of Hopkins statistic. The red and blue lines are the empirical density curves of 1000 Hopkins statistics calculated with exponents $d = 1$ (red) and $d = 3$ (blue). The black line is the theoretical distribution of the Hopkins statistic. The red line is very far away from the black line and shows that calculating Hopkins statistic with exponent $d = 1$ is incorrect.

- 1982: Cross and Jain (1982) generalized Hopkins statistic for X of any dimension $d = D$ as $\sum u_i^d / \sum(u_i^d + w_i^d)$. This formula was also used by Zeng and Dubes (1985a), Dubes and Zeng (1987), and Banerjee and Dave (2004).
- 1990: Lawson and Jurs (1990) and Jurs and Lawson (1990) give the formula for Hopkins statistic as $\sum u_i / \sum(u_i + w_i)$, but used ordinary distances instead of squared distances. Perhaps this was a result of misunderstanding the formula in Diggle, Besag, and Gleaves (1976).
- 2015: The R function `hopkins()` in the **clustertend** package (YiLan and RuTong 2015 version 1.4) cited Lawson and Jurs (1990) and used also used the exponent $d = 1$.
- 2022: The new function `hopkins()` in the **hopkins** package (Wright 2022 version 1.0) uses the general exponent $d = D$ as found in Cross and Jain (1982).

3 Simulation study for the distribution of Hopkins statistic

Having identified the confusion in the statistical literature, we now ask the question, "Does it matter what value of d is used in the exponent?" In a word, "yes".

According to Cross and Jain (1982), under the null hypotheses of no structure in the data, the distribution of the Hopkins statistic is Beta(m,m) where m is the number of rows sampled in X. This distribution can be verified in a simple simulation study:

1. Generate a matrix X with 100 rows (events) and $D = 3$ columns, filled with random uniform numbers. (This is the assumption of no spatial structure in a 3D hypercube.)
2. Sample $m = 10$ events and also generate 10 new uniform points.
3. Calculate Hopkins statistic with exponents $d = 1$ (incorrect value).
4. Calculate Hopkins statistic with exponents $d = 3$ (correct value).
5. Repeat 1000 times.
6. Compare the empirical density curves of the two different methods to the Beta(m,m) distribution.

In Figure 1:

- The black curve is the density of Beta(10,10).
- The red curve is the density of Hopkins statistic when $d = 1$ is used in the calculation (incorrect).
- The blue curve is the density of Hopkins statistic when $d = 3$ (the number of columns in X) is used (correct).

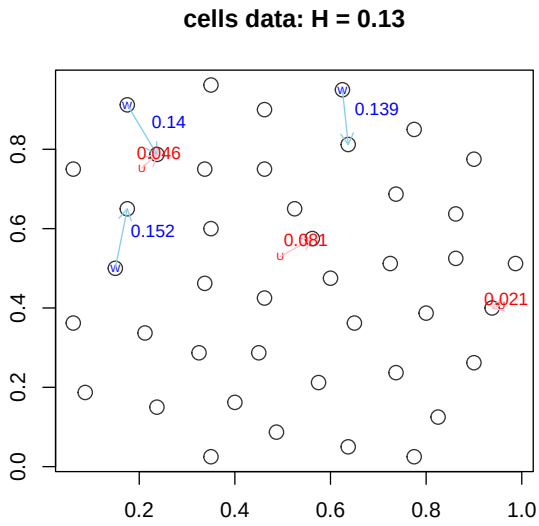


Figure 2: An example of how Hopkins statistic is calculated with systematically-spaced data. The black circles are the events of the 'cells' data. Each blue 'W' represents a randomly-chosen event. Each blue arrow points from a 'W' to the nearest-neighboring event. Each red 'U' is a new, randomly-generated point. Each red arrow points from a 'U' to the nearest-neighboring event. The numbers are the length of the arrows. In systematically-spaced data, red arrows tend to be shorter than blue arrows.

The empirical density of the blue curve is similar to the theoretical distribution shown by the black line. The empirical density of the red curve is clearly dissimilar. The distribution of Hopkins statistic with $d = 1$ is clearly incorrect (except in trivial cases where X has only 1 column). One more thing to note about the graph is that the blue curve is slightly flatter than the theoretical distribution shown in black. This is not accidental, but is caused by edge effects of the sampling region and will be discussed in a later section.

4 Examples

The first three examples in this section are adapted from Gastner (2005). The datasets are available in the spatstat.data package (Baddeley, Turner, and Rubak 2021). A modified version of the `hopkins()` function was written for this paper to show how the Hopkins statistic is calculated (inspired by Figure 1 of Lawson and Jurs (1990)). In order to minimize the amount of over-plotting, only $m = 3$ sampling points are used for these examples. In each figure, 3 of the existing events in X are chosen at random and a light-blue arrow is drawn to the nearest neighbor in X. In addition, 3 points are drawn uniformly in the plotting region and a light-red arrow is drawn to the nearest neighbor in X. The colored numbers are the lengths of the arrows.

Example 1: Systematically-spaced data

The `cells` data represent the centers of mass of 42 cells from insect tissue. The scatterplot of the data in Figure 2 shows that events are systematically spaced as nearly far apart as possible. Because the data is two-dimensional, Hopkins statistics is calculated as the sum of the squared distances u_i^2 divided by the sum of the squared distances $u_i^2 + w_i^2$:

```
(.046^2 + .081^2 + .021^2) /
  ( (.046^2 + .081^2 + .021^2) + (.152^2 + .14^2 + .139^2) )
```

```
#> [1] 0.1281644
```

The `hopkins()` function returns the same value:

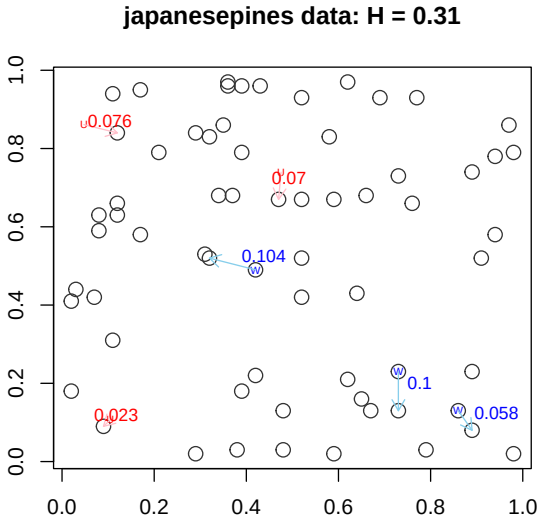


Figure 3: An example of how Hopkins statistic is calculated with randomly-spaced data. The black circles are the events of the 'japanesepines' data. Each blue 'W' represents a randomly-chosen event. Each blue arrow points from a 'W' to the nearest-neighboring event. Each red 'U' is a new, randomly-generated point. Each red arrow points from a 'U' to the nearest-neighboring event. The numbers are the length of the arrows. In randomly-spaced data, red arrows tend to be similar in length to blue arrows.

```
set.seed(17)
hopkins(cells, m=3)

#> [1] 0.1285197
```

The value of the Hopkins statistic in this calculation is based on only $m = 3$ events and will have sizable sampling error. To reduce the sampling error, a larger sample size can be used up to approximately 10% of the number of events. To reduce sampling error further while maintaining the independence assumption of the sampling in calculating Hopkins statistic, repeated samples can be drawn. Here we use the idea of Gastner (2005) to calculate Hopkins statistic 100 times and then calculate the mean and standard deviation for the 100 values of Hopkins statistic, which in this case are 0.21 and 0.06. This value of the statistic is quite a bit lower than 0.5 and indicates the events are spaced more evenly than purely-random events (p-value 0.05).

Example 2: Randomly-spaced data

The `japanesepines` data contains the locations of 65 Japanese black pine saplings in a square 5.7 meters on a side. The plot of the data in Figure 3 is an example of data in which the events are randomly spaced.

The value of Hopkins statistic using 3 events and points is:

```
(.023^2+.076^2+.07^2) /
  ((.023^2+.076^2+.07^2) + (.104^2+.1^2+.058^2))

#> [1] 0.3166596
```

The mean and standard deviation of the 100 Hopkins statistics are 0.48 and 0.12. The value of the statistic is close to 0.5 and indicates no evidence against a random distribution of data (p-value 0.9).

Example 3: Clustered data

The `redwood` data are the coordinates of 62 redwood seedlings in a square 23 meters on a side. The plot in Figure 4 shows events that exhibit clustering. The value of Hopkins statistic for the plot is:

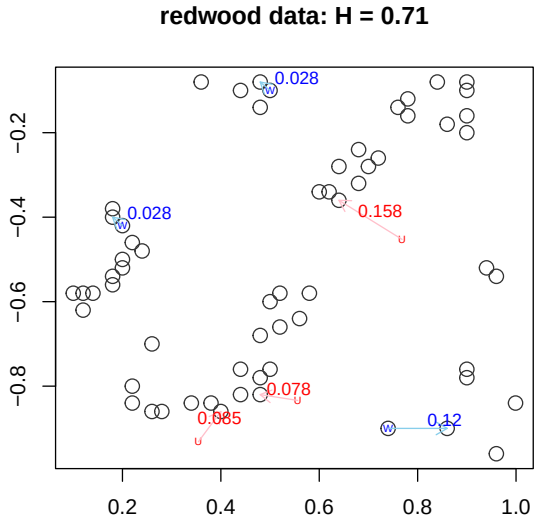


Figure 4: An example of how Hopkins statistic is calculated with clustered data. The black circles are the events of the 'redwood' data. Each blue 'W' represents a randomly-chosen event. Each blue arrow points from a 'W' to the nearest-neighboring event. Each red 'U' is a new, randomly-generated point. Each red arrow points from a 'U' to the nearest-neighboring event. The numbers are the length of the arrows. In clustered data, red arrows tend to be longer in length than blue arrows.

```
(.085^2+.078^2+.158^2) /
  ((.085^2+.078^2+.158^2) + (.028^2+.028^2+.12^2))

#> [1] 0.7056101
```

The mean and standard deviation of the 100 Hopkins statistics are 0.79 and 0.13. The value of the statistic is much higher than 0.5, which indicates that the data are somewhat clustered (p-value 0.03).

Example 4

Adolfsson, Ackerman, and Brownstein (2017) provide a review of various methods of detecting clusterability. One of the methods they considered was Hopkins statistic, which they calculated using 10% sampling. They evaluated the clusterability of nine R datasets by calculating Hopkins statistic 100 times and then reporting the proportion of time that Hopkins statistic exceeded the appropriate beta quantile. We can repeat their analysis and calculate Hopkins statistic for both $d = 1$ dimension and $d = D$ dimensions, where D is the number of columns for each dataset.

In Table 1:

- Column 1 is the name of the R dataset.
- Column 2 is the number of observations n.
- Column 3 is the number of dimensions D.
- Column 4 is the proportion of 100 times that Hopkins statistic is significant as reported by Adolfsson, Ackerman, and Brownstein (2017).
- Columns 5 and 6 use the **hopkins** package. Column 5 is the proportion of 100 times that Hopkins statistic with exponent $d = 1$ and column 6 is the proportion of 100 times that Hopkins statistic with exponent $d = D$ is significant.

Since the `Adolfsson` and `Hopkins1` columns are similar (within sampling variability), it appears that Adolfsson, Ackerman, and Brownstein (2017) used Hopkins statistic with $d = 1$ as the exponent. This would be expected if they had used the **clustertend** package (YiLan and RuTong 2015 version 1.4) to calculate Hopkins statistic.

For a few of the datasets, there is substantial disagreement between the last two columns. For example, the `swiss` data is significantly clusterable 41% of the time according to Adolfsson, Ackerman, and Brownstein (2017), but 94% of the time when using Hopkins statistic with exponent $d = D$. A

Table 1: In this table, 'dataset' is the R dataset name, 'n' is the number of rows in the data, 'D' is the number of columns in the data, 'Adolfsson' is the the proportion of 100 times that Hopkins statistic was significant as appearing in the paper by Adolfsson et al. (2017), 'Hopkins1' is the proportion of 100 times that Hopkins statistic was significant when calculated with the exponent $d = 1$ (similar to the 'clustertend' package), and 'HopkinsD' is the proportion of 100 times that Hopkins statistic was significant when calculated with the exponent $d = D$. Since the 'Adolfsson' and 'Hopkins1' columns are similar (within samling variation), it appears that Adolfsson et al. (2017) used the 'clustertend' package to calculate Hopkins statistic.

dataset	n	D	Adolfsson	Hopkins1	HopkinsD
faithful	272	2	1.00	1.00	1.00
iris	150	5	1.00	1.00	1.00
rivers	141	1	0.92	0.89	0.90
swiss	47	6	0.41	0.25	0.94
attitude	30	7	0.00	0.00	0.59
cars	50	2	0.19	0.23	0.68
trees	31	3	0.18	0.22	0.71
USJudgeRatings	43	12	0.69	0.53	1.00
USArrests	50	4	0.01	0.00	0.56

scatterplot of the `swiss` data in Figure 5 shows that the data are strongly non-random, which agrees with the 94%.

Similarly, the `trees` data is significantly clusterable 18% of the time according to the `Adolfsson` column, but 71% of the time according to `HopkinsD`. The scatterplot in Figure 6 shows strong non-random patterns, which agrees with the 71%

Scatterplot matrices of the `swiss`, `attitude`, `cars`, `trees`, and `USArrests` datasets can be found in Brownstein, Adolfsson, and Ackerman (2019). Each scatterplot matrix shows at least one pair of the variables with notable correlation and therefore the data are not randomly-distributed, but rather are clustered. For each of these datasets, the proportion of times `Hopkins1` is significant is less than 0.5, but the proportion of times `HopkinsD` is significant is greater than 0.5. The `HopkinsD` statistic is accurately detecting the presence of clusters in these datasets.

5 Correcting for edge effects

In the `cells`, `japanesepines` and `redwood` examples above, it is possible or even probable that there are additional events outside of the sampling frame that contains the data. The sampling frame thus has the effect of cutting off potential nearest neighbors from consideration. If the distribution of the data can be assumed to extend beyond the sampling frame and if the shape of the sampling frame can be viewed as a hypercube, then edge effects due to the sampling frame can be corrected by using a torus geometry that wraps edges of the sampling frame around to the opposite side (Li and Zhang 2007). To see an illustration of this, look again at the plot of the `japanesepines` data in Figure 3. The randomly-generated event U in the upper left corner is a distance of 0.076 away from the nearest event. However, if the left edge of the plot is wrapped around an imaginary cylinder and connected to the right edge of the plot, then the nearest neighbor is the event in the upper-right corner at coordinates (0.97, 0.86).

To see what effect the torus geometry has on the distribution of the Hopkins statistic, consider the following simulation. We generate $n = 100$ events uniformly in a $D = 5$ dimension unit cube and sample $m = 10$ events to calculate the value of Hopkins statistic using both a simple geometry and a torus geometry. Repeat these steps $B = 1000$ times. The calculation of the nearest neighbor using a torus geometry is computationally more demanding than using a simple geometry, especially as the number of dimensions D increases, so the use of parallel computing can reduce the computing time linearly according to the number of processors used. As a point of reference, this small simulation study was performed in less than 1 minute on a reasonably-powerful laptop with 8 cores using the **doParallel** package (Microsoft Corporation and Weston 2020). We found that $B = 1000$ provided results that were stable, regardless of the seed value for the random number generation in the simulations.

In Figure 7:

- The black curve is the density of Beta(10,10).
- The blue curve is the empirical density of the 1000 values of Hopkins statistic calculated using a *simple* geometry.
- The green curve is the empirical density of the 1000 values of Hopkins statistic calculated using a *torus* geometry.

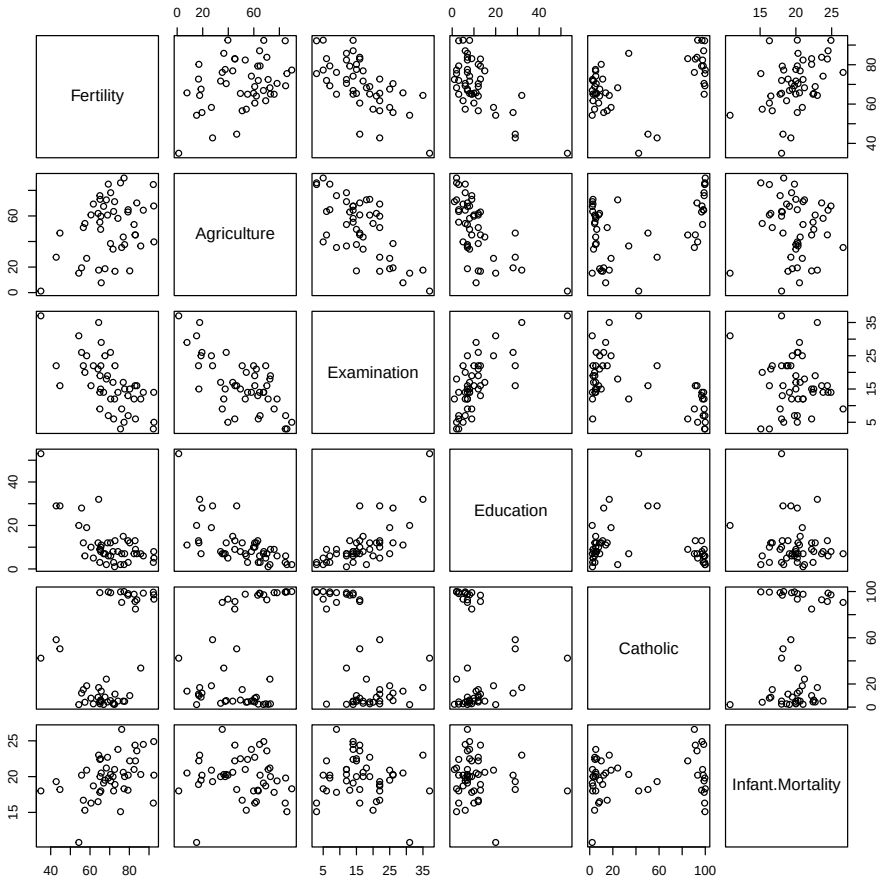


Figure 5: Pairwise scatterplots of the R dataset 'swiss'. The meaning of the variables is not important here. Because some panels show a lack of spatial randomness of the data, we would expect Hopkins statistic to be significant.

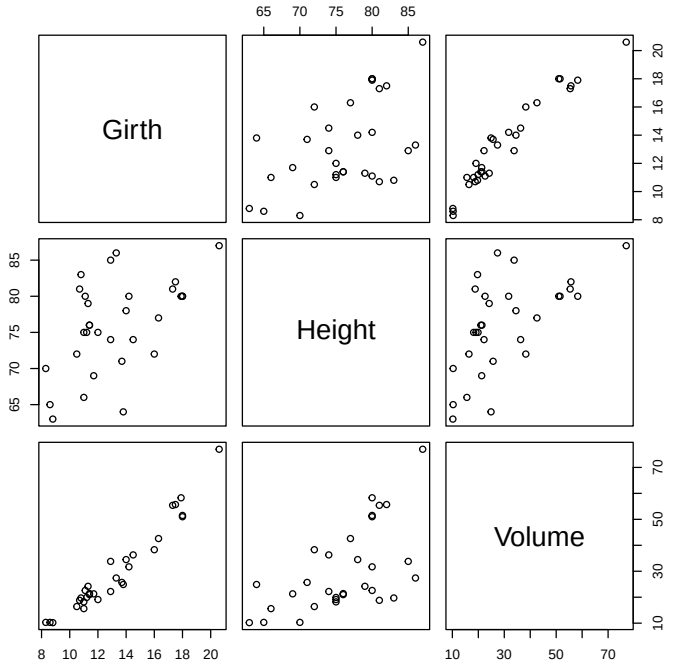


Figure 6: Pairwise scatterplots of the R dataset 'trees'. The data are 'Girth', 'Height', and 'Volume' of 31 black cherry trees. Because all panels show a lack of spatial randomness of the data, we would expect Hopkins statistic to be significant.

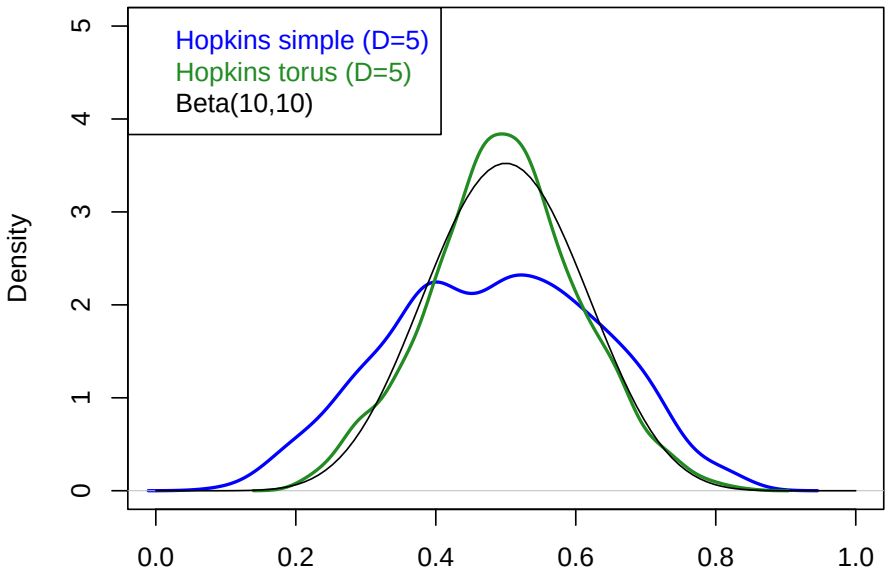


Figure 7: Results of a simulation study considering how the spatial geometry affects Hopkins statistic. The thin black line is the theoretical distribution of Hopkins statistic. The blue and green lines are the empirical density curves of 1000 Hopkins statistics calculated with simple geometry (blue) and torus geometry (green). Calculating Hopkins statistic with a torus geometry aligns closely to the theoretical distribution.

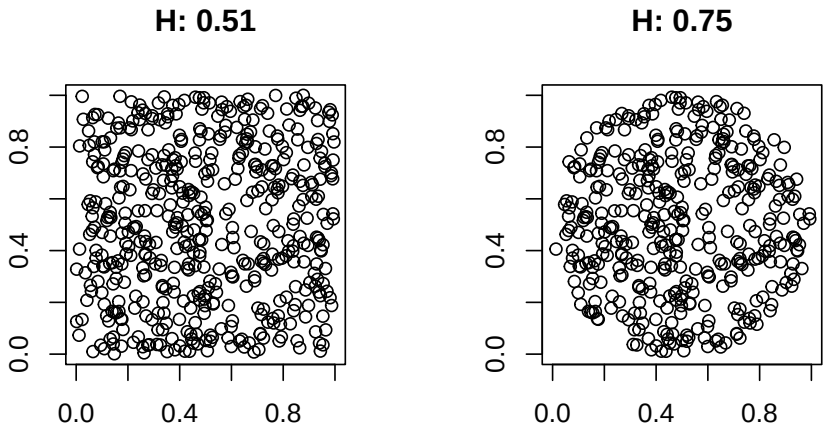


Figure 8: The left figure shows 250 points simulated randomly in a unit square. As expected, the value of Hopkins statistic is close to 0.5. The right figure shows the same points, but only those inside a unit-diameter circle. The value of Hopkins statistic H is much larger than 0.5. Although both figures depict spatially-uniform points, the square shape of the sampling frame affects the value of Hopkins statistic.

When using a torus geometry to correct for edge effects in this example, the empirical distribution of Hopkins statistic is remarkably close to its theoretical distribution. In contrast, when a simple geometry is used, the empirical distribution of Hopkins statistic is somewhat flattened with heavier tails. The practical result is that when no edge correction is used, the Hopkins statistic is more likely to deviate from 0.5 and therefore more likely to suggest the data is not uniformly distributed. This erroneous interpretation is a greater risk as the number of dimensions D increases and edge effects become more pronounced

6 Sampling frame effects

Another practical problem affecting the correct use and interpretation of Hopkins statistic has to do with the shape of the sampling frame. Consider the example data in Figure 8. On the left side, there were 250 random events simulated in a 2-dimensional unit square. On the right side, the same data are used, but have been subset to keep only the events inside a unit-diameter circle. For both figures, Hopkins statistic was calculated 100 times with 10 events sampled each time.

On the left side, both the bounding box and the actual sampling frame are the unit square. The median of 100 Hopkins statistics is 0.51, providing no evidence against random distribution. On the right side, the actual sampling frame of the data is a unit-circle, but the Hopkins statistic still uses the unit square (for generating new points in U) and the median Hopkins statistic is 0.75, indicating clustering of the data *within the sampling frame*, even though the distribution of the data was generated uniformly. A few more examples of problems related to the sampling frame can be found in Smith and Jain (1984).

To consider the problem with the sampling frame on real data, refer again to the `trees` data in Figure 6. Because trees usually grow both in height and girth at the same time, it would be unexpected to find tall trees with narrow girth or short trees with large girth. Also, since the volume is a function of the girth and height, it is correlated with those two variables. In the scatterplot of girth versus volume, it would be nearly impossible to find points in the upper left or lower right corner of the square. From a biological point of view, the sampling frame cannot be shaped like a square and the null hypothesis of uniform distribution of data is violated *a priori*, which means the distribution of Hopkins statistic does not follow a Beta(m,m) distribution.

7 A protocol for using Hopkins statistic

Because Hopkins statistic is not hard to calculate and is easy to interpret, yet can be misused (as shown in the previous sections), we propose a protocol for using Hopkins statistic. The protocol simply asks the practitioner to consider the following five questions before calculating Hopkins statistic.

1. **Is the number of events n $>$ 100 and the number of randomly-sampled events at most 10% of n?** This is recommended by Cross and Jain (1982).
2. **Is spatial randomness of the events even possible?** If the events are known or suspected to be correlated, this violates the null hypothesis of spatial uniformity, and may also mean that the sampling frame is not shaped like a hypercube.
3. **Could nearest-neighbor events have occurred outside the boundary of the sampling frame?** If yes, it may be appropriate to calculate nearest-neighbor distances using a torus geometry.
4. **Is the sampling frame non-rectangular?** If yes, then be extremely careful with the use of Hopkins statistic in how points are samples from U.
5. **Is the dimension of the data much greater than 2?** Edge effects are more common in higher dimensions.

The important point of this protocol is to raise awareness of potential problems. We leave it to the practitioner to decide what do with the answers to these questions.

8 Conclusion

The statistical literature regarding Hopkins statistic is filled with confusion about how to calculate the statistic. Some publications have erroneously used the exponent $d = 1$ in the formula for Hopkins statistic and this error has propagated into much statistical software and led to incorrect conclusions. To remedy this situation, the R package **hopkins** (Wright 2022) provides a function `hopkins()` that calculates Hopkins statistic using the general exponent $d = D$ for D-dimensional data. The function can use simple geometry for fast calculations or torus geometry to correct for edge effects. Using this function, we show that the distribution of Hopkins statistic calculated with the general exponent $d = D$ aligns closely with the theoretical distribution of the statistic. Because inference with Hopkins statistic can be trickier than expected, we introduce a protocol of five questions to consider when using Hopkins statistic.

Alternative versions of Hopkins statistic have been examined by Zeng and Dubes (1985b), Rotondi (1993), Li and Zhang (2007). Other methods of examining multivariate uniformity of data have been considered by Smith and Jain (1984), Yang and Modarres (2017), and Petrie and Willemain (2013).

9 Acknowledgements

Thanks to Deanne Wright for bringing the confusion about Hopkins statistic to our attention. Thanks to Vanessa Windhausen and Deanne Wright for reading early drafts of this paper and to Dianne Cook for reviewing the final version. Thanks to Wong (2013) for the **pdist** package for fast computation of nearest neighbors and thanks to Northrop (2021) for the **donut** package for nearest neighbor search on a torus.

References

Adolfsson, Andreas, Margareta Ackerman, and Naomi C Brownstein. 2017. "To Cluster, or Not to Cluster: How to Answer the Question." *Proceedings of Knowledge Discovery from Data, Halifax, Nova Scotia, Canada, August 13-17 (TKDD '17)*. https://maya-ackerman.com/wp-content/uploads/2018/09/clusterability2017.pdf.

Baddeley, Adrian, Rolf Turner, and Ege Rubak. 2021. *spatstat.data: Datasets for 'Spatstat' Family*. https://CRAN.R-project.org/package=spatstat.data.

Banerjee, Amit, and Rajesh N Dave. 2004. "Validating Clusters Using the Hopkins Statistic." In *2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No. 04CH37542)*, 1:149–53. IEEE. https://doi.org/10.1109/FUZZY.2004.1375706.

Brownstein, Naomi C, Andreas Adolfsson, and Margareta Ackerman. 2019. "Descriptive Statistics and Visualization of Data from the R Datasets Package with Implications for Clusterability." *Data in Brief* 25: 104004. https://doi.org/10.1016/j.dib.2019.104004.

Cross, G R, and A K Jain. 1982. "Measurement of Clustering Tendency." In *Theory and Application of Digital Control*, 315–20. https://doi.org/10.1016/B978-0-08-027618-2.50054-1.

Diggle, Peter J, Julian Besag, and J Timothy Gleaves. 1976. "Statistical Analysis of Spatial Point Patterns by Means of Distance Methods." *Biometrics*, 659–67. https://doi.org/10.2307/2529754.

Dubes, Richard C, and Guangzhou Zeng. 1987. "A Test for Spatial Homogeneity in Cluster Analysis." *Journal of Classification* 4 (1): 33–56. https://doi.org/10.1007/BF01890074.

Gastner, Michael T. 2005. "Spatial Distributions: Density-Equalizing Map Projections, Facility Location, and Two-Dimensional Networks." PhD thesis, Univ. Michigan. https://hdl.handle.net/2027.42/125368.

Hopkins, Brian, and John Gordon Skellam. 1954. "A New Method for Determining the Type of Distribution of Plant Individuals." *Annals of Botany* 18 (2): 213–27. https://doi.org/10.1093/oxfordjournals.aob.a083391.

Jurs, Peter C, and Richard G Lawson. 1990. "Clustering Tendency Applied to Chemical Feature Selection." *Drug Information Journal* 24 (4): 691–704. https://doi.org/10.1177/216847909002400405.

Lawson, Richard G, and Peter C Jurs. 1990. "New Index for Clustering Tendency and Its Application to Chemical Problems." *Journal of Chemical Information and Computer Sciences* 30 (1): 36–41. https://doi.org/10.1021/ci00065a010.

Li, Fasheng, and Lianjun Zhang. 2007. "Comparison of Point Pattern Analysis Methods for Classifying the Spatial Distributions of Spruce-Fir Stands in the North-East USA." *Forestry* 80 (3): 337–49. https://doi.org/10.1093/forestry/cpm010.

Microsoft Corporation, and Steve Weston. 2020. *doParallel: Foreach Parallel Adaptor for the 'Parallel' Package*. https://CRAN.R-project.org/package=doParallel.

Northrop, Paul J. 2021. *donut: Nearest Neighbour Search with Variables on a Torus*. https://CRAN.R-project.org/package=donut.

Petrie, Adam, and Thomas R. Willemain. 2013. "An Empirical Study of Tests for Uniformity in Multidimensional Data." *Computational Statistics & Data Analysis* 64: 253–68. https://doi.org/10.1016/j.csda.2013.02.013.

Rotondi, Renata. 1993. "Tests of Randomness Based on the k-NN Distances for Data from a Bounded Region." *Probability in the Engineering and Informational Sciences* 7: 557–69. https://doi.org/10.1017/S0269964800003132.

Smith, Stephen P, and Anil K Jain. 1984. "Testing for Uniformity in Multidimensional Data." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 1: 73–81. https://doi.org/10.1109/TPAMI.1984.4767477.

Wong, Jeffrey. 2013. *pdist: Partitioned Distance Function*. https://CRAN.R-project.org/package=pdist.

Wright, Kevin. 2022. *hopkins: Calculate Hopkins Statistic for Clustering*. https://CRAN.R-project.org/package=hopkins.

Yang, Mengta, and Reza Modarres. 2017. "Multivariate Tests of Uniformity." *Statistical Papers* 58: 627–39. https://doi.org/10.1007/s00362-015-0715-x.

YiLan, Luo, and Zeng RuTong. 2015. *clustertend: Check the Clustering Tendency*. https://CRAN.R-project.org/package=clustertend.

Zeng, Guangzhou, and Richard C Dubes. 1985a. "A Comparison of Tests for Randomness." *Pattern Recognition* 18 (2): 191–98. https://doi.org/10.1016/0031-3203(85)90043-3.

———. 1985b. "A Test for Spatial Randomness Based on k-NN Distances." *Pattern Recognition Letters* 3 (2): 85–91. https://doi.org/10.1016/0167-8655(85)90013-3.

Kevin Wright
Corteva Agriscience
6805 NW 62nd Ave
Johnston, IA 50131
ORCiD: 0000-0002-0617-8673
kw.stat@gmail.com

Multivariate Subgaussian Stable Distributions in R

by Bruce J. Swihart, John P. Nolan

Abstract We introduce and showcase **mvpd** (an acronym for *m*ulti*v*ariate *p*roduct *d*istributions), a package that uses a product distribution approach to calculating multivariate subgaussian stable distribution functions. The family of multivariate subgaussian stable distributions are elliptically contoured multivariate stable distributions that contain the multivariate Cauchy and the multivariate normal distribution. These distributions can be useful in modeling data and phenomena that have heavier tails than the normal distribution (more frequent occurrence of extreme values). Application areas include log returns for stocks, signal processing for radar and sonar data, astronomy, and hunting patterns of sharks.

1 Introduction

Multivariate subgaussian stable distributions are the elliptically contoured subclass of general multivariate stable distributions. To begin a brief introduction to multivariate subgaussian stable distributions, we start with univariate stable distributions which may be more readily familiar and accessible. Univariate stable distributions are a flexible family and have four parameters, α, β, γ, and δ, and at least eleven parameterizations (!) which has led to much confusion (Nolan, 2020). Here we focus on the 1-parameterization of the Nolan style. Location is controlled by δ, scale by $\gamma \in (0, \infty)$, while $\alpha \in (0, 2]$ and $\beta \in [-1, 1]$ can be considered shape parameters. Being a location-scale family, a "standard" stable distribution will be when $\gamma = 1$ and $\delta = 0$. A solid introduction to univariate stable distributions can be found in the recent textbook *Univariate Stable Distributions* (Nolan, 2020) and its freely available Chapter 1 online (`https://edspace.american.edu/jpnolan/stable/`).

Univariate symmetric stable distributions are achieved by setting the skew parameter $\beta = 0$, which gives symmetric distributions that are bell-shaped like the normal distribution. A way to remember that these are called subgaussian is to see that as $\alpha \in (0, 2]$ increases from 0 it looks more and more normal until it **is** normal for $\alpha = 2$ (Figure 1). The *sub* in subgaussian refers to the tail behavior in that the rate of decrease in the tails is *less* than that of a gaussian – note how the tails are above the gaussian for $\alpha < 2$ in Figure 1. Equivalently, as α decreases, the tails get heavier. A notable value of α for subgaussian distributions is $\alpha = 1$ which is the Cauchy distribution. The Cauchy and Gaussian distribution are most well-known perhaps because they have closed-form densities, which all other univariate symmetric stable distributions lack.

Therefore, numerically computing the densities is especially important for application. For univariate stable distributions, there is open-source software to compute modes, densities, distributions, quantiles and random variates, including a number of R packages (**stabledist**, **stable**, **libstableR**, for example – see CRAN Task View: Probability *Distributions* for more).

As generalizations of the univariate stable distribution, multivariate stable distributions are a very broad family encompassing many complicated distributions (e.g. support in a cone, star shaped level curves, etc.). A subclass of this family is the multivariate subgaussian stable distributions. Multivariate subgaussian stable distributions are symmetric and elliptically contoured. Similar to the aforementioned univariate symmetric stable distributions, the value $\alpha = 2$ is the multivariate gaussian and $\alpha = 1$ is the multivariate Cauchy. Being that they are elliptically contoured and symmetric makes them applicable to finance where joint returns have an (approximately) elliptical joint distribution (Nolan, 2020). Signal processing, such as with radar and sonar data, tasks itself with filtering impulsive noise from a signal of interest and linear filters in the presence of extreme values tend to underperform, whereas using multivariate stable distributions have been fruitful (Tsakalides and Nikias, 1998; Nolan, 2013). The (multivariate) Holtsmark distribution is a multivariate subgaussian stable distribution ($\alpha = 1.5$) that has applications in astronomy, astrophysics, and plasma physics. Lévy flights, which are random walks with steps having a specific type of multivariate subgaussian stable distribution, are used to model interstellar turbulence as well as hunting patterns of sharks (Boldyrev and Gwinn, 2003; Sims et al., 2008).

For multivariate subgaussian stable distributions, the parameter α is a scalar as in the univariate family, while δ (location) becomes a d-dimensional vector and the analogue for the scale parameter is a $d \times d$ shape matrix Q. The shape matrix Q needs to be semi-positive definite and is neither a covariance matrix nor covariation matrix. An introduction to multivariate subgaussian stable distributions can be found in Nolan (2013).

Including **mvpd**, the focus of this paper, if one wanted `R` functions to interact with multivariate

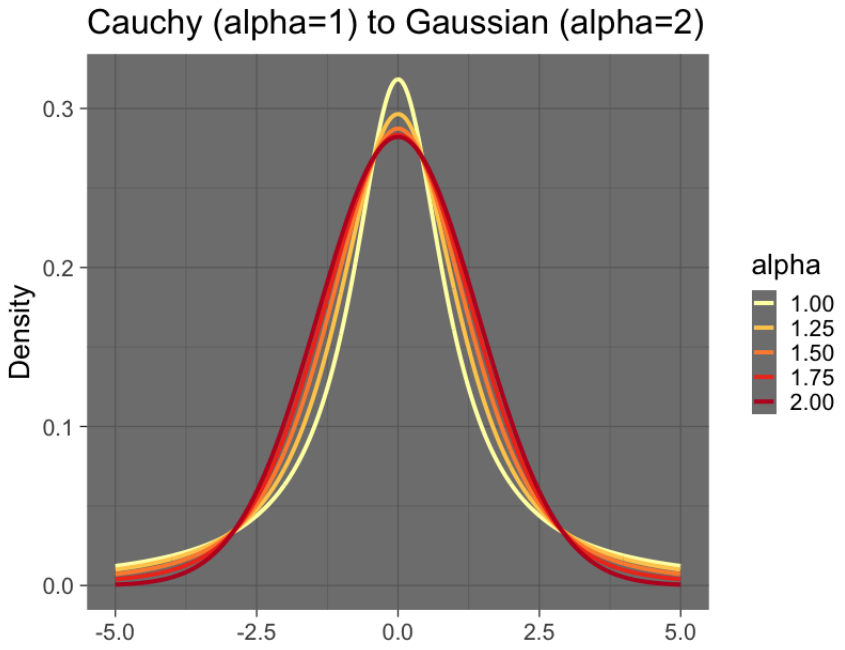


Figure 1: The lower the value of alpha, the heavier the tails. The superimposed standard univariate subgaussian densities for selected values of alpha are displayed. Commonly known distributions include the Cauchy (alpha=1) and normal (alpha=2).

Functionality	`alphastable`	`stable`	`mvpd`
random variates	✓	✓	✓
parameter estimation	✓	✓	✓
density	✓	✓	✓
cumulative distribution (monte carlo)	x	✓	✓
cumulative distribution (integrated)	x	x	✓
multivariate subgaussian stable	✓	✓	✓
multivariate independent stable	x	✓	x
multivariate isotropic stable	x	✓	x
multivariate discrete-spectral-measure stable	x	✓	x

Table 1: Summary of functionality among `R` packages. Note: The `stable` package referenced is not the one on CRAN – it is proprietary software produced by the company Robust Analysis.

subgaussian stable distributions they have three `R` package options. These packages are compared in Table 1 and detailed below:

- **alphastable** provides random univariate and multivariate generation, density calculation, and parameter fitting (albeit for modest sample sizes) via an EM algorithm method (Teimouri et al., 2018, 2019).
- **stable** provides support for all stable univariate distributions and multivariate subgaussian stable distributions. Other cases are handled, to varying degrees, such as isotropic, independent, and spectral measure. For the purposes of this paper, we will note that the **stable** package provides random variate generation, density calculation, parameter fitting, distribution calculations via Monte Carlo methods for multivariate subgaussian stable distributions $2 \leq d \leq 100$. The **stable** package is developed by Robust Analysis and is available for purchase or through a software grant at `http://www.robustanalysis.com/`. It is distinct from the univariate **stable** package on CRAN authored by Jim Lindsey.
- **mvpd** provides random variate generation, density calculation, parameter fitting, distribution function calculations via Monte Carlo methods, as well as an integrated method for distribution calculations that allows tolerance specification.

While the lack of a tractable density and distribution function impedes directly calculating multivariate subgaussian stable distributions, it is possible to represent them in terms of a product distribution for which each of the two distributions in the product has known numerical methods

developed and deployed (in R packages on CRAN). This paper utilizes this approach. The next section covers some product distribution theory.

2 Product distribution theory

This section reviews some known results of *product distributions* and describes our notation. Allow univariate positive random variable A with density $f_A(x)$ and d-dimensional random variable G to have density $f_G(\boldsymbol{x})$ and distribution function $F_G(\boldsymbol{v}, \boldsymbol{w}) = P(\boldsymbol{v} < \boldsymbol{x} \leq \boldsymbol{w})$. Consider the $d-$dimensional product $H = A^{1/2}G$. From standard product distribution theory we know the density f_H is represented by 1-dimensional integral:

$$f_H(\boldsymbol{x}) \quad = \quad \int_0^\infty f_B(u) f_G(\boldsymbol{x}/u) \frac{1}{|u|^d} du\,, \tag{1}$$

where $B = A^{1/2}$ so that $f_B(x) := 2x f_A(x^2)$. Consequently the distribution function F_H (with lower bound $\boldsymbol{v}$ and upper bound $\boldsymbol{w}$) of the r.v. H is represented by

$$\begin{aligned} F_H(\boldsymbol{v}, \boldsymbol{w}) &= \int_0^\infty f_B(u) \int_{v_1}^{w_1} \cdots \int_{v_d}^{w_d} f_G(\boldsymbol{t}/u) \frac{1}{|u|^d} dt_1 \ldots dt_d du\,, \qquad (2) \\ &= \int_0^\infty f_B(u) \int_{v_1/u}^{w_1/u} \cdots \int_{v_d/u}^{w_d/u} f_G(\boldsymbol{t}) dt_1 \ldots dt_d du\,, \\ &= \int_0^\infty f_B(u) F_G(\boldsymbol{v}/u, \boldsymbol{w}/u) du. \qquad (3) \end{aligned}$$

Take note of the representation in (1) and (3). From a practical standpoint, if we have a (numerical) way of calculating f_A, f_G, and F_G we can calculate f_H and F_H. Different choices can be made for f_A, f_G, and F_G in this general setup. The choices required for multivariate subgaussian stable distributions are covered in the following Implementation section.

Multivariate elliptically contoured stable distributions

From Nolan (2013), H is a d-dimensional subgaussian stable distribution if A is a positive univariate stable distribution

$$A \sim S\left(\frac{\alpha}{2}, 1, 2\cos\left(\frac{\pi\alpha}{4}\right)^{\left(\frac{2}{\alpha}\right)}, 0; 1\right)$$

and G is a d-dimensional multivariate normal $G \sim MVN(0, Q)$, where the shape matrix Q is positive semi-definite. This corresponds to Example 17 in Hamdan (2000).

3 Implementation

Using the aforementioned theory of product distributions, we can arrive at functions for a multivariate subgaussian stable density and distribution function thanks to established functions for univariate stable and multivariate normal distributions. A key package in the implementation of multivariate subgaussians in R is **mvtnorm** (Genz et al., 2020; Genz and Bretz, 2009). In the basic product-distribution approach of **mvpd**, f_G and F_G are `mvtnorm::dmvnorm` and `mvtnorm::pmvnorm` respectively. Allow the density of A, f_A (to be numerically calculated in R) using `stable::dstable` or `libstableR::stable_pdf` (del Val et al., 2017). Presented as pseudo-code:

- $f_A(x, \alpha) :=$ `libstableR::stable_pdf(`x`, pars =` $\left(\frac{\alpha}{2}, 1, 2\cos\{\frac{\pi\alpha}{4}\}^{\left(\frac{2}{\alpha}\right)}, 0\right)$`; pm = 1)`
- $f_B(x, \alpha) := 2x f_A(x^2, \alpha)$
- $f_H(\boldsymbol{x}, \alpha, Q) = \int_0^\infty f_B(u; \alpha) \times$ `mvtnorm::dmvnorm(`$\boldsymbol{x}/u$`, sigma =` $Q) \frac{1}{|u|^d} du$
- $F_H(\boldsymbol{v}, \boldsymbol{w}, \alpha, Q) = \int_0^\infty f_B(u; \alpha) \times$ `mvtnorm::pmvnorm(lower =` $\boldsymbol{v}/u$`, upper =` $\boldsymbol{w}/u$`, sigma =` $Q) du$

The (outermost) univariate integral is numerically evaluated with `stats::integrate`.

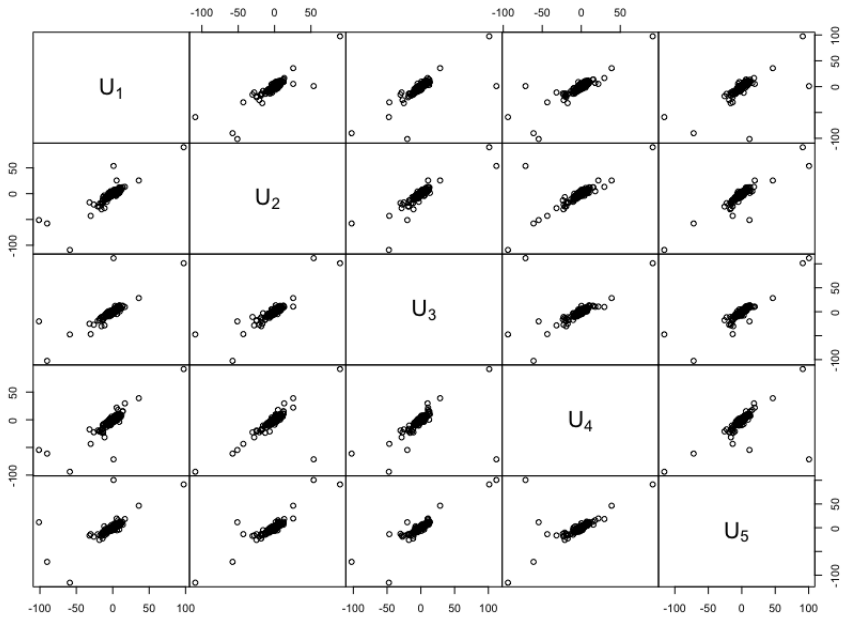


Figure 2: A lattice of scatterplots of 5,000 draws from a 5-dimensional subgaussian stable distribution, showing the pairwise relations. The outliers from the cloud in each plot demonstrate the heavy tails.

4 Quick start

We present an outline of the `[dpr]mvss` (*m*ulti*v*ariate *s*ubgaussian *s*table) functions, and walk through the code in the subsequent sections. As an overview, we generate 5000 5-dimensional subgaussian variates with $\alpha = 1.7$ and an "exchangeable" shape matrix using `rmvss`. We then recover the parameters with an illustrative call to `fit_mvss`. We can calculate the density (`dmvss`) at the center of the distribution and get a quick estimate of the distribution between -2 and 2 for each member of the 5-dimensional variate using `pmvss_mc`. We investigate a refinement of that quick distribution estimate using `pmvss`.

5 Random variates generation with `rmvss`

We'll generate 5000 5-dimensional subgaussian random variates with a specified α and shape matrix. They are pictured in Figure 2. In the next section we will fit a distribution to these.

```
R> library(mvpd)
## reproducible research sets the seed
R> set.seed(10)
## specify a known 5x5 shape matrix
R> shape_matrix <- structure(c(1, 0.9, 0.9, 0.9, 0.9,
                               0.9, 1, 0.9, 0.9, 0.9,
                               0.9, 0.9, 1, 0.9, 0.9,
                               0.9, 0.9, 0.9, 1, 0.9,
                               0.9, 0.9, 0.9, 0.9, 1),
                               .Dim = c(5L, 5L))
## generate 5000 5-dimensional random variables
## with alpha = 1.7 and shape_matrix
R> X <- mvpd::rmvss(n = 5000, alpha = 1.7, Q = shape_matrix)
## plot all pairwise scatterplots (Figure 2)
R> copula::pairs2(X)
```

The ability to simulate from a distribution is useful for running simulations to test different scenarios about the phenomena being modeled by the distribution, as well as in this case, to generate a dataset with a known shape matrix and alpha to show our fitting software (next section) can recover these parameters. Our quick start code begins with generating a dataset from a known alpha and shape matrix. However, often a practitioner might start with a dataset from which parameters are

estimated and then random samples can be generated from the distribution specified with those parameters to learn more about the data generating distribution and the behavior of the phenomena.

6 Fitting a multivariate subgaussian distribution with `fit_mvss`

If you have data in a $n \times d$ matrix $\mathbf{X}$ and want to fit a d-dimensional multivariate subgaussian distribution to those data, then `fit_mvss` will return estimates of the parameters using the method outlined in Nolan (2013). The method involves fitting univariate stable distributions for each column and assessing the resulting α, β and δ parameters. The column-wise α estimates should be similar and the column-wise β estimates close to 0. This column-wise univariate fitting is carried out by `libstableR::stable_fit_mle2d(W,parametrization = 1L)` and the off diagonal elements can be found due to the properties of univariate stable distributions (see Nolan (2013)). For your convenience, the univariate column-wise estimates of α, β, γ and δ are returned in addition to the raw estimate of the shape matrix and the nearest positive definite shape matrix (as computed by `Matrix::nearPD` applied to the raw estimate).

```
## take X from previous section and estimate
## parameters for the data generating distribution
R> fitmv <- mvpd::fit_mvss(X)
R> fitmv
$univ_alphas
[1] 1.698617 1.708810 1.701662 1.696447 1.699372

$univ_betas
[1] -0.02864287 -0.04217262 -0.08444540 -0.06569907 -0.03228573

$univ_gammas
[1] 1.016724 1.000151 1.008055 1.012017 1.002993

$univ_deltas
[1] -0.03150732 -0.06525291 -0.06528644 -0.07730645 -0.04539796

$mult_alpha
[1] 1.700981

$mult_Q_raw
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0337276 0.9034599 0.8909654 0.8937814 0.8647089
[2,] 0.9034599 1.0003026 0.9394846 0.9072368 0.8535091
[3,] 0.8909654 0.9394846 1.0161748 0.8929937 0.9037467
[4,] 0.8937814 0.9072368 0.8929937 1.0241777 0.9281714
[5,] 0.8647089 0.8535091 0.9037467 0.9281714 1.0059955

$mult_Q_posdef
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0337276 0.9034599 0.8909654 0.8937814 0.8647089
[2,] 0.9034599 1.0003026 0.9394846 0.9072368 0.8535091
[3,] 0.8909654 0.9394846 1.0161748 0.8929937 0.9037467
[4,] 0.8937814 0.9072368 0.8929937 1.0241777 0.9281714
[5,] 0.8647089 0.8535091 0.9037467 0.9281714 1.0059955
```

An alternative for fitting this distribution is `alphastable::mfitstab.elliptical(X,1.70,shape_matrix,rep(0,5))` and takes 8 minutes (and requires initial values for alpha, the shape matrix, and delta). This analysis with `fit_mvss(X)` took under 2 seconds. For a run of `n=1e6,d=20`, `fit_mvss` scales well, taking 60 minutes.

Once the distribution has been fitted, `fitmv$mult_alpha`, `fitmv$mult_Q_posdef`, and `fitmv$univ_deltas`, can be used as the `alpha`, `Q`, and `delta` arguments, respectively, in calls to `dmvss` to calculate densities and `pmvss_mc` or `pmvss` to calculate probabilities. They could also be passed to `rmvss` to generate random variates for simulations.

7 Density calculations with `dmvss`

We can calculate the density at the center of the distribution.

```
## density calculation
R> mvpd::dmvss(x = fitmv$univ_deltas,
+              alpha = fitmv$mult_alpha,
+              Q = fitmv$mult_Q_posdef,
+              delta = fitmv$univ_deltas)[1]
$value
[1] 0.1278952
```

8 Distribution calculation by Monte Carlo with `pmvss_mc`

The method of calculating the distribution by Monte Carlo relies on the ability to produce random variates quickly and then calculate what proportion of them fall within the specified bounds. To generate multivariate subgaussian stable variates, a scalar A is drawn from

$$\texttt{libstableR::stable\_rnd}(n, \texttt{pars =} \left(\frac{\alpha}{2}, 1, 2\cos\{\frac{\pi\alpha}{4}\}^{\left(\frac{2}{\alpha}\right)}, 0\right); \texttt{pm = 1})$$

and then the square-root of A multiplied by a draw G from

$$\texttt{mvtnorm::rmvnorm}(n, \text{sigma} = Q).$$

This allows for quick calculations but to increase precision requires generating larger number of random variates. For instance, if we wanted the distribution between -2 and 2 for each dimension, we could generate 10,000 random variates and then see how many of them fall between the bounds. It looks like 6,820 variates were within the bounds:

```
## first-run of pmvss_mc
R> mvpd::pmvss_mc(lower = rep(-2,5),
+                 upper = rep( 2,5),
+                 alpha = fitmv$mult_alpha,
+                 Q = fitmv$mult_Q_posdef,
+                 delta = fitmv$univ_deltas,
+                 n = 10000)
[1] 0.6820
```

We run it again and the answer changes:

```
## second-run of pmvss_mc
R> mvpd::pmvss_mc(lower = rep(-2,5),
+                 upper = rep( 2,5),
+                 alpha = fitmv$mult_alpha,
+                 Q = fitmv$mult_Q_posdef,
+                 delta = fitmv$univ_deltas,
+                 n = 10000)
[1] 0.6742
```

With the Monte Carlo method, precision is not specified and no error is calculated. The next section introduces how to use the integrated distribution function F_H from product theory and specify precision.

9 Distribution function calculation via integration with `pmvss`

There are three inexact entities involved in the distribution calculation F_H as found in `pmvss`: the numerically calculated F_G, the numerically calculated f_A, and the outer numerical integration.

The outer integral by `integrate` assumes the integrand is calculated without error, but this is not the case. See the supplementary materials section "Thoughts on error propagation in `pmvss`" for justification and guidance for specifying the values of `abs.tol.si`, `abseps.pmvnorm`, and `maxpts.pmvnorm`. The first of these three arguments is passed to the `abs.tol` argument of `stats::integrate` and

controls the absolute tolerance of the numerically evaluated outer 1-dimensional integral. The remaining two are passed to `maxpts` and `abseps` of `mvtnorm::GenzBretz` and control the accuracy of `mvtnorm::pmvnorm`.

Briefly, our experience suggests that to be able to treat `abs.tol.si` as the error of the result, `abseps.pmvnorm` should be 1e-2 times smaller than the specified `abs.tol.si` which may require a multiple of the default 25000 default of `maxpts.pmvnorm` – which will lead to more computational intensity and longer computation times as demonstrated below (as conducted on Macbook Intel Core i7 chip with 2.2 GHz):

```
## abs.tol.si    abseps.pmnvorm   maxpts         Time
## 1e-01         1e-03            25000
## 1e-02         1e-04            25000*10        3 sec
## 1e-03         1e-05            25000*100      22 sec
## 1e-04         1e-06            25000*1000      4 min
## 1e-05         1e-07            25000*10000    26 min
## 1e-06         1e-08            25000*85000   258 min
```

With this in mind, the output from the Quick Start code is:

```
## precision specified pmvss
R> mvpd::pmvss(lower = rep(-2,5),
+              upper = rep( 2,5),
+              alpha = fitmv$mult_alpha,
+              Q = fitmv$mult_Q_posdef,
+              delta = fitmv$univ_deltas,
+              abseps.pmvnorm = 1e-4,
+              maxpts.pmvnorm = 25000*10,
+              abs.tol.si = 1e-2)[1]
$value
[1] 0.6768467
```

Both `pmvss` and `pmvss_mc` take infinite limits. Since `pmvss_mc` calculates how many random variates H_i, $i \in \{1, \ldots, n\}$ are within the bounds, `pmvss` might be preferred to `pmvss_mc` when calculating the tails of the distribution, unless n is made massively large.

10 Speed and accuracy trials

We provide a sense of accuracy and computational time trade-offs with a modest simulation experiment (Figure 3, see supplementary materials for code). Estimating these distributions is inherently difficult – difficult in the sense that expecting accuracy farther out than the 5th decimal place for distribution functions is unreasonable. Therefore, we will define our "gold standard" targets for accuracy evaluation as the numerical density produced by Robust Analysis' `dstable` integrated by `cubature::hcubature()` with tolerance `tol=1e-5`.

We will time three functions using `bench::mark()` in different scenarios. The first function is Robust Analysis' `pmvstable.MC()` (abbreviated as RAMC, below) and the other two are `mvpd::pmvss_mc()` (abbreviated as PDMC, below) and `mvpd::pmvss()` (abbreviated as PD, below). Fixing $\alpha = 1.7$ and dimension $d = 4$, the different test scenarios will involve a low level of pairwise association vs. a high level in a shape matrix of the form:

- $Q_{exch} = \begin{bmatrix} 1 & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho \\ \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & 1 \end{bmatrix}$ for $\rho = 0.1$ and $\rho = 0.9$.

We calculate the distributions in the hypercube bounded by (-2,2) in all four dimensions. The gold standard for the $\rho = 0.1$ case was 0.5148227 and 0.7075104 for the $\rho = 0.9$ case. The numerical integration of the former took 3 minutes whereas the latter took 1 hour – which portends that higher associations involve more computational difficulty. We back-calculated the number of samples needed to give the methods involving Monte Carlo (RAMC and PDMC) a 95% CI width that would fall within 0.001 and 0.0001 of the gold standard, and display the scatter plots of estimate and computational time in (Figure 3).

From Figure 3, some high-level conclusions can be drawn: higher pairwise associations require more computational resources and time, increasing the precision requires more computational resources and time, and sometimes the Monte Carlo methods are faster than PD, sometimes not. PD seems to be quite precise and possibly underestimating the gold standard.

Of course, we cannot test every possible instance of alpha and shape matrices for all d dimensions, integration limits, and specified precision. In our experience, the computational intensity is an interplay between alpha, the integration limits, the shape matrix structure, delta, and the requested precision. We provide the code that we used for our simulation study and encourage the readers who need to explore these issues for their particular integral to edit the code accordingly.

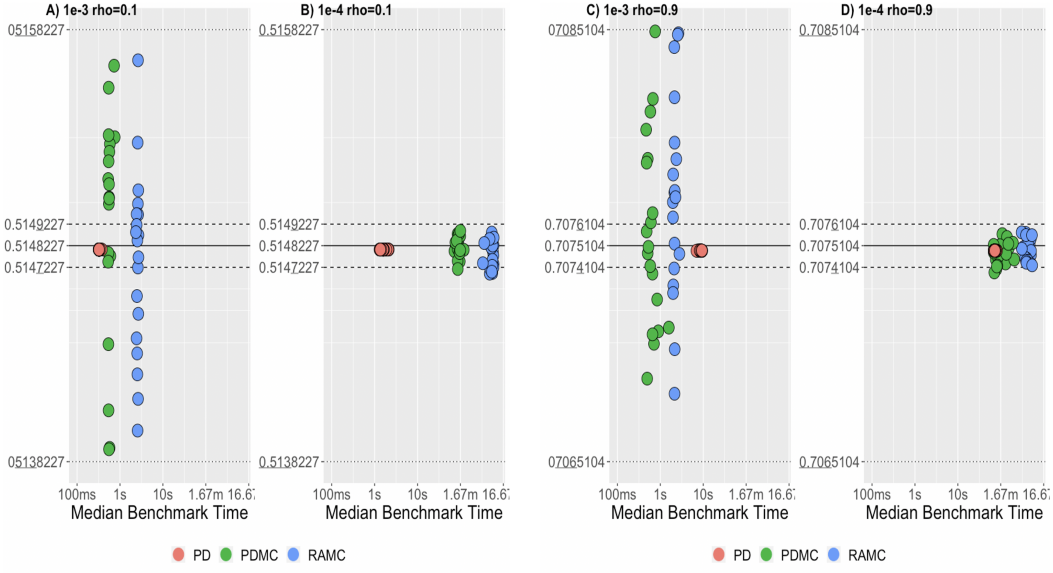


Figure 3: Speed and accuracy of distribution calculations. Consider a multivariate subgaussian stable distribution of $d = 4$, $\alpha = 1.7$, with limits of integration being (-2,2) for each dimension. In panels A) and B) we have an exchangeable shape matrix with $\rho = 0.1$, and specified precision of 1e-3 and 1e-4 for `pmvss` (PD), respectively. Analogously, panels C) and D) display results for an exchangeable shape matrix with $\rho = 0.9$. Concurrently in each panel, are the results for Robust Analysis' `pmvstable.MC` (RAMC) and `pmvss_mc` (PDMC) with enough simulated variates to produce a 95 CI width that matches the precision. Each point is an independent call and the calculated distribution is on the Y-axis vs the median benchmark time on the X-axis. There are 20 calls per function per scenario. The dotted line is the 1e-3 boundary of the gold standard and the dashed line is the 1e-4 boundary.

11 Bonus: faster distribution calculations via a modified `QRSVN` algorithm

Insight: multivariate student's t distributions are a product distribution

The derivation of the univariate student's t distribution is commonly motivated with a ratio of two quantities each involving random variables: a standard normal Z in the numerator and a $V \sim \chi^2(\nu)$ in the denominator:

$$T_\nu = \frac{Z}{\sqrt{V/\nu}} = Z\sqrt{\frac{\nu}{V}},$$

but what often is left out of the instruction is that $A_\nu = \frac{\nu}{V} \sim IG\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$ has an inverse-gamma distribution, where $X \sim IG(r,s)$ with rate r, shape s, and density $f(x;r,s) = \frac{r^s}{\Gamma(s)} x^{(-s-1)} e^{-r/x}$. This implies we equivalently have a product distribution of the type $T_\nu = A_\nu^{1/2} Z$. This notion holds for the multivariate case as well, where for $G \sim MVN(0,Q)$ as before and A_ν is an inverse-gamma with $r = s = \nu/2$ then $H_\nu = A_\nu^{1/2} G$ is a d-dimensional student's t distribution with ν degrees of freedom and covariance matrix Q. This corresponds to Example 16 in in Hamdan (2000), and is equivalent to the 'chi-normal' (χ-Φ) formulation in Genz and Bretz (2002) (earning the namesake 'χ' due to the fact that $V \sim \chi^2(\nu) \implies \sqrt{V} \sim \chi(\nu)$). For $d \geq 4$, the QRSVN algorithm (`https://www.math.wsu.edu/faculty/genz/software/fort77/mvtdstpack.f`) is used in `mvtnorm::pmvt`. The reordering and rotational methodology that makes `pmvtnorm::pmvt` so fast is independent of the part that generates $\sqrt{1/A_v}$ random variates. This means that if one replaced $\sqrt{1/A_v}$ with $\sqrt{1/A}$ variates, `mvtnorm::pmvt` would produce *not* multivariate student's t distributions but **multivariate subgaussian stable distributions**. We implement a modified QRSVN algorithm for multivariate subgaussian stable distributions in a separate package, **mvgb** in honor of Genz and Bretz.

Implementation of `mvgb::pmvss`

Generating random variates of A requires two independent uniform random variates, and only one of which is Quasi-Random in our implementation. Regardless, this modified QRSVN approach enables the potential advantage of the rotation of the distribution and the reordering of integration limits. The takeaway is, that for similar precision, `mvgb::pmvss` may be much faster than `mvpd::pmvss`, such as 10 seconds vs 500 seconds for 4 digits of precision in the following example:

```
R> set.seed(321)
R> library(mvgb)
R> tictoc::tic()
## probability calculated by mvgb takes about 10 seconds
R> gb_4digits <-
+   mvgb::pmvss(lower = rep(-2,5),
+               upper = rep( 2,5),
+               alpha = fitmv$mult_alpha,
+               Q = fitmv$mult_Q_posdef,
+               delta = fitmv$univ_deltas,
+               abseps = 1e-4,
+               maxpts = 25000*350)
R> tictoc::toc()
9.508 sec elapsed
> gb_4digits
[1] 0.6768
## now calculate same probability with similar precision
## in mvpd
R> tictoc::tic()
## probability calculated by mvpd takes about 10 MINUTES
R> pd_4digits <-
+   mvpd::pmvss(lower = rep(-2,5),
+               upper = rep( 2,5),
+               alpha = fitmv$mult_alpha,
+               Q = fitmv$mult_Q_posdef,
+               delta = fitmv$univ_deltas,
+               abseps.pmvnorm = 1e-6,
+               maxpts.pmvnorm = 25000*1000,
+               abs.tol.si = 1e-4)
R> tictoc::toc()
518.84 sec elapsed
R> pd_4digits[1]
[1] 0.6768
```

Although currently on CRAN, we include `mvgb::pmvss` here as a proof-of-concept and as an area of future work. More research is needed into its computational features and accuracy, and this is encouraged by promising preliminary results. Additionally, more research may be warranted for other R package methodologies that use a multivariate Gaussian, Cauchy, or Holtsmark distribution to generalize to a multivariate subgaussian stable distribution (a helpful reviewer suggested generalizing the multivariate distributions as used in **fHMM** (Oelschläger and Adam, 2021) and generalizing the normally mixed probit model in **RprobitB**). For more about elliptically contoured multivariate distributions in general, consult Fang and Anderson (1990); Fang et al. (2018).

Acknowledgements

This work utilized the computational resources of the NIH HPC Biowulf cluster (http://hpc.nih.gov). We thank Robust Analysis for providing their **stable** `R` package via a software grant.

Bibliography

S. Boldyrev and C. R. Gwinn. Lévy model for interstellar scintillations. *Physical review letters*, 91(13): 131101, 2003. [p293]

J. R. del Val, F. Simmross-Wattenberg, and C. Alberola-López. libstable: Fast, parallel, and high-precision computation of α-stable distributions in R, C/C++, and MATLAB. *Journal of Statistical Software*, 78(1):1–25, 2017. doi: 10.18637/jss.v078.i01. [p295]

K.-T. Fang and T. W. Anderson. *Statistical inference in elliptically contoured and related distributions*. Allerton Press, 1990. [p301]

K.-T. Fang, S. Kotz, and K. W. Ng. *Symmetric multivariate and related distributions*. Chapman and Hall/CRC, 2018. [p301]

A. Genz and F. Bretz. Comparison of methods for the computation of multivariate t probabilities. *Journal of Computational and Graphical Statistics*, 11(4):950–971, 2002. [p300]

A. Genz and F. Bretz. *Computation of Multivariate Normal and t Probabilities*. Lecture Notes in Statistics. Springer-Verlag, Heidelberg, 2009. ISBN 978-3-642-01688-2. [p295]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, and T. Hothorn. *mvtnorm: Multivariate Normal and t Distributions*. The organization, 2020. URL https://CRAN.R-project.org/package=mvtnorm. R package version 1.1-0. [p295]

H. N. Hamdan. *PhD Thesis: Characterizing and approximating scale mixtures*. American University, 2000. [p295, 300]

J. P. Nolan. Multivariate elliptically contoured stable distributions: theory and estimation. *Computational Statistics*, 28(5):2067–2089, 2013. [p293, 295, 297]

J. P. Nolan. *Univariate stable distributions*. Springer, 2020. [p293]

L. Oelschläger and T. Adam. Detecting bearish and bullish markets in financial time series using hierarchical hidden markov models. *Statistical Modelling*, page 1471082X211034048, 2021. [p301]

D. W. Sims, E. J. Southall, N. E. Humphries, G. C. Hays, C. J. Bradshaw, J. W. Pitchford, A. James, M. Z. Ahmed, A. S. Brierley, M. A. Hindell, et al. Scaling laws of marine predator search behaviour. *Nature*, 451(7182):1098–1102, 2008. [p293]

M. Teimouri, S. Rezakhah, and A. Mohammadpour. Parameter estimation using the em algorithm for symmetric stable random variables and sub-gaussian random vectors. *Journal of Statistical Theory and Applications*, 17(3):439–461, 2018. [p294]

M. Teimouri, A. Mohammadpour, and S. Nadarajah. *alphastable: Inference for Stable Distribution*, 2019. URL https://CRAN.R-project.org/package=alphastable. R package version 0.2.1. [p294]

P. Tsakalides and C. L. Nikias. *Deviation from Normality in Statistical Signal Processing: Parameter Estimation*. Springer Science & Business Media, 1998. [p293]

Bruce J. Swihart
National Institutes of Health
National Institute of Allergy and Infectious Diseases
5601 Fishers Lane Baltimore MD, 20852
United States of America
(ORCiD 0000-0002-4216-9942)
bruce.swihart@nih.gov

John P. Nolan
American University
Department of Math & Statistics
4400 Mass. Ave, Washington DC 20016
United States of America
(ORCiD 0000-0002-9669-382X)
jpnolan@american.edu

Bioconductor Notes, Autumn 2022

by Bioconductor Core Developer Team

Abstract We discuss the release of Bioconductor 3.16, along with educational activities and general project news.

1 Introduction

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. The project has entered its twentieth year, with funding for core development and infrastructure maintenance secured through 2025 (NIH NHGRI 2U24HG004059). Additional support is provided by NIH NCI, Chan-Zuckerberg Initiative, National Science Foundation, Microsoft, and Amazon. In this news report, we give some details about the software and data resource collection, infrastructure for building, checking, and distributing resources, core team activities, and some new initiatives.

2 Software

Bioconductor 3.16 was released on 2 November, 2022. It is compatible with R 4.2 and consists of 2183 software packages, 416 experiment data packages, 909 up-to-date annotation packages, 28 workflows, and 3 books. Books are built regularly from source and therefore fully reproducible; an example is the community-developed Orchestrating Single-Cell Analysis with Bioconductor. The Bioconductor 3.16 release announcement includes descriptions of 71 new software packages, 9 new data experiment packages, 2 new annotation packages, and updates to NEWS files for many additional packages.

3 Core team updates

- New developer Robert Shear of the Department of Data Science at Dana-Farber Cancer Institute has joined the Bioconductor Core Developer Team.
- Robert is joined by long-term core members Lori Kern of Roswell Park Comprehensive Cancer Center, Marcel Ramos of CUNY and Roswell, Herv'e Pages of Fred Hutchinson Cancer Research Center, Jennifer Wokaty of CUNY, and Alex Mahmoud at Channing Division of Network Medicine.

4 Educational activities and resources

Engagement with The Carpentries

In August 2022, Bioconductor joined The Carpentries. Details and opportunities for receiving training on teaching are discussed in this blog post. We are currently inviting applications to become a Bioconductor Carpentries instructor through this form and particularly encourage people who could teach underserved communities in their local languages to apply.

Three lessons are under development in the Carpentries incubator: Introduction to data analysis with R and Bioconductor, RNA-seq analysis with Bioconductor and The Bioconductor project. We welcome any contributions, feedback or testing of the material.

Anyone is welcome to join the #education-and-training channel in Bioconductor Slack or the monthly Bioconductor Teaching Committee meetings to learn more.

YES for CURE

The Dana-Farber/Harvard Cancer Center Young Empowered Scientists program included a module on cancer data science for Summer 2022 participants. Materials presented are assembled at a pkgdown site; contact Vince Carey for information on an interactive deployment of these materials.

5 Using Bioconductor

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 provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key 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 workspace (sign up) for extended technical discussion.
- The F1000Research Bioconductor gateway for peer-reviewed Bioconductor workflows as well as conference contributions.
- The Bioconductor YouTube channel includes recordings of keynote and talks from recent conferences including BioC2022, EuroBioC2022, and BiocAsia2021, in addition to video recordings of training courses.
- Our package submission repository for open technical review of new packages.

Upcoming and recently completed conferences are browsable at our events page.

The Technical and 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
Channing Division of Network Medicine
Mass General Brigham
Harvard Medical School, Boston, MA

Department of Data Science
Dana-Farber Cancer Institute
Harvard Medical School, Boston, MA

Biostatistics and Bioinformatics
Roswell Park Comprehensive Cancer Center, Buffalo, NY

Fred Hutchinson Cancer Research Center, Seatlle, WA

CUNY Graduate School of Public Health, New York, NY

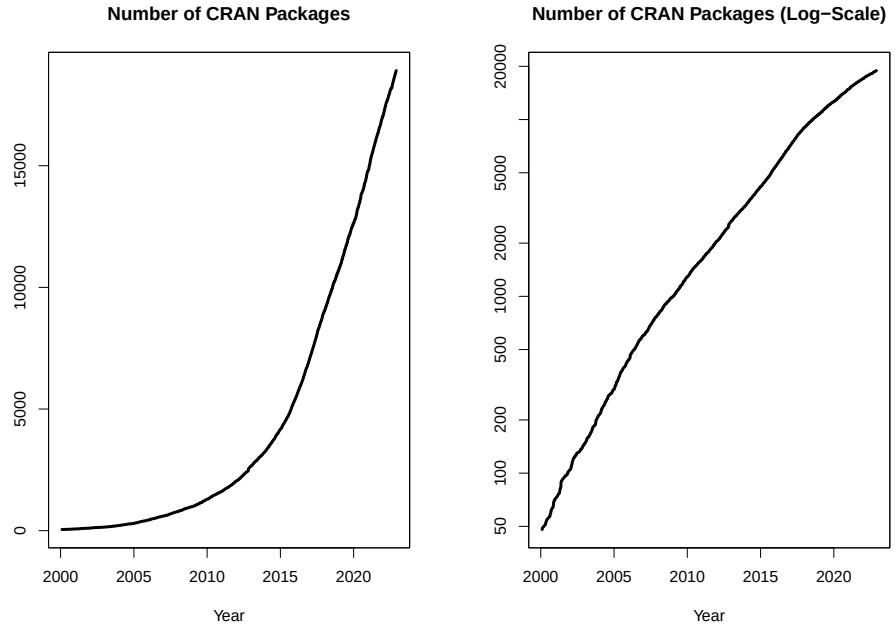
Bioconductor Core Developer Team
Dana-Farber Cancer Institute, Roswell Park Comprehensive Cancer Center, City University of New York, Fred Hutchinson Cancer Research Center, Mass General Brigham

Changes on CRAN

2022-07-01 to 2022-10-31

by Kurt Hornik, Uwe Ligges and Achim Zeileis

In the past 4 months, 664 new packages were added to the CRAN package repository. 154 packages were unarchived, 353 were archived and 6 had to be removed. The following shows the growth of the number of active packages in the CRAN package repository:



On 2022-10-31, the number of active packages was around 18730.

CRAN package submissions

From September 2022 to October 2022 CRAN received 5227 package submissions. For these, 9429 actions took place of which 6070 (64%) were auto processed actions and 3359 (36%) 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	1335	1211	926	0	0	1560	513	525
manual	1250	35	325	232	39	1054	353	71

These include the final decisions for the submissions which were

	archive	publish
auto	1283 (25.1%)	1239 (24.3%)
manual	1222 (23.9%)	1361 (26.7%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

CRAN mirror security

Currently, there are 100 official CRAN mirrors, 81 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

There are three new task views:

Agricultural Science Maintained by Julia Piaskowski, Adam Sparks, and Janet Williams.

Mixed, Multilevel, and Hierarchical Models in R Maintained by Ben Bolker, Julia Piaskowski, Emi Tanaka, Phillip Alday, and Wolfgang Viechtbauer.

Phylogenetics Maintained by William Gearty, Brian O'Meara, Jacob Berv, Gustavo A. Ballen, Diniz Ferreira, Hilmar Lapp, Lars Schmitz, Martin R. Smith, Nathan S. Upham, and Jonathan A. Nations.

Currently there are 42 task views (see `https://cran.r-project.org/web/views/`), with median and mean numbers of CRAN packages covered 102 and 115, respectively. Overall, these task views cover 4015 CRAN packages, which is about 21% of all active CRAN packages.

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`