

The Journal

Volume 6/2, December 2014

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

Contents

Editorial 3

Contributed Research Articles

Coordinate-Based Meta-Analysis of fMRI Studies with R 5

Automatic Conversion of Tables to LongForm Dataframes 16

Prinsimp 27

phaseR: An R Package for Phase Plane Analysis of Autonomous ODE Systems. 43

Flexible R Functions for Processing Accelerometer Data, with Emphasis on NHANES
2003–2006 52

Applying **spartan** to Understand Parameter Uncertainty in Simulations 63

ngspatial: A Package for Fitting the Centered Autologistic and Sparse Spatial Gener-
alized Linear Mixed Models for Areal Data 81

sgof: An R Package for Multiple Testing Problems 96

bshazard: A Flexible Tool for Nonparametric Smoothing of the Hazard Function . . . 114

SMR: An R Package for Computing the Externally Studentized Normal Midrange
Distribution 123

Farewell’s Linear Increments Model for Missing Data: The **FLIM** Package 137

MVN: An R Package for Assessing Multivariate Normality 151

qmethod: A Package to Explore Human Perspectives Using Q Methodology. 163

gset: An R Package for Exact Sequential Test of Equivalence Hypothesis Based on
Bivariate Non-Central t -Statistics. 174

News and Notes

Conference Report: R in Insurance 2014 185


Conference Report: Polish Academic R User Meeting 187

R Foundation News 190

Changes on CRAN 192

Changes in R 224

News from the Bioconductor Project 227

The  Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are licensed under the Creative Commons Attribution 3.0 Unported license (CC BY 3.0, <http://creativecommons.org/licenses/by/3.0/>).

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

Editor-in-Chief:

Deepayan Sarkar

Editorial Board:

Hadley Wickham, Bettina Grün and Michael Lawrence

Editor Help Desk:

Uwe Ligges

Editor Book Reviews:

G. Jay Kerns

Department of Mathematics and Statistics

Youngstown State University

Youngstown, Ohio 44555-0002

USA

gkerns@ysu.edu

R Journal Homepage:

<http://journal.r-project.org/>

Email of editors and editorial board:

firstname.lastname@R-project.org

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

Editorial

by *Deepayan Sarkar*

On behalf of the editorial board, I am pleased to publish Volume 6, Issue 2 of the R Journal.

The first article in this issue presents a case study illustrating the use of R for analyzing fMRI data. The second article introduces the innovative **TableToLongForm** package that tries to automate the preprocessing of commonly encountered hierarchical tables that are often difficult to convert into machine-readable form. The remaining articles describe a variety of R packages that make new methods available to the R community: **Prinsimp** to study interesting structures in low variability principal components, **phaseR** for autonomous ODE Systems, **accelerometry** and **nhanesaccel** for processing accelerometer data, **spartan** to analyze simulation results, **ngspatial** for analyzing areal data, **sgof** for multiple testing problems, **bshazard** for survival analysis, **SMR** which implements a set of distribution functions, **FLIM** for longitudinal studies with missing data, **MVN** for assessing multivariate normality, **qmethod** for studying perspectives and attitudes using Q methodology, and **gset** for exact sequential tests of equivalence hypotheses. In addition to the usual updates, the News and Notes section also contains two conference reports. I hope you enjoy the issue.

With the end of the year, it is also time for a refresh of the editorial board. Hadley Wickham is leaving the board after a four-year term. As with everything he turns his attention to, Hadley has considerably streamlined the behind-the-scenes operations of the Journal, making life easier for the rest of us. We welcome Roger Bivand who is joining the editorial board in his place. Publishing this issue is my last act as Editor-in-Chief, with Bettina Grün taking over the job for the next year.

Deepayan Sarkar

deepayan.sarkar@r-project.org

Coordinate-Based Meta-Analysis of fMRI Studies with R

by *Andrea Stocco*

Abstract This paper outlines how to conduct a simple meta-analysis of neuroimaging foci of activation in R. In particular, the first part of this paper reviews the nature of fMRI data, and presents a brief overview of the existing packages that can be used to analyze fMRI data in R. The second part illustrates how to handle fMRI data by showing how to visualize the results of different neuroimaging studies in a so-called orthographic view, where the spatial distribution of the foci of activation from different fMRI studies can be inspected visually.

Functional MRI (fMRI) is one of the most important and powerful tools of neuroscientific research. Although not as commonly used for fMRI analysis as some specific applications such as SPM (Friston et al., 2006), AFNI (Cox and Hyde, 1997), or FSL (Smith et al., 2004), R does provide several packages that can be employed in neuroimaging research. These packages deal with a variety of topics, ranging from reading and manipulating fMRI datasets, to implementing sophisticated statistical models.

The goal of this paper is to provide a brief introduction to fMRI analysis, and the various R packages that can be used to carry it out. As an example, it will show how to use simple R commands to read fMRI images and plot results from previous studies, which can then be visually compared. This is a special form of meta-analysis, and a common way to compare results from the existing literature.

A brief introduction to fMRI data analysis

Before getting into the details of the R code, it is important to have a clear picture of what type of data and analyses are common in fMRI research. An fMRI study consists of a set of 3D images that capture brain activity at fixed intervals of time while participants are performing a given task inside an MRI scanner. The interval between image acquisitions is known as *repetition time* or *TR*, and is typically 2 seconds. Each image is made of thousands of almost-cubic elements called *voxels*. The size of a voxel (typically on the order of $3 \times 3 \times 3$ mm) represents the lower limit of the spatial resolution of an image. Not all the voxels are acquired at the same time: each image is typically created by acquiring each horizontal 2D plane (or “slice”) in serial order. Thus, voxels belonging to different slices are acquired at different times within the same 2-second TR.

The MRI scanner captures brain “activity” only indirectly, by measuring the so-called Blood Oxygen-Level Dependent (BOLD) signal—that is, the amount of oxygenated blood present in every position in space. The BOLD signal is known to vary as a function of neural activity. Specifically, the BOLD response to an increase of neural activity at time $t = 0$ follows a specific time course, which is known as the hemodynamic response function $h(t)$, and commonly modeled as the difference between two gamma functions (Worsley et al., 2002):

$$h(t) = \left(\frac{t}{d_1}\right)^{a_1} \times e^{-\frac{t-d_1}{b_1}} - c \left(\frac{t}{d_2}\right)^{a_2} \times e^{-\frac{t-d_2}{b_2}}$$

Figure 1 illustrates the typical shape of the hemodynamic response function, as returned by the function `fmri.stimulus` of the `fmri` package (Polzehl and Tabelow, 2007; Tabelow and Polzehl, 2011) using the following code.

```
library(fmri)
t <- seq(0, 30, 0.1) # from 0 to 30 secs, in increments of 100ms
h <- fmri.stimulus(301, durations = c(0), # h(t) for instantaneous event at t=0,
                 onsets = c(1), rt = 0.1), # Sampled at TR = 0.1 secs
plot(t, h, type = "l", lwd = 2, col = "red",
     xlab = "Time (secs)", ylab = "h(t)", main = "Hemodynamic Response Function")
```

Note how the function peaks only about 6 seconds *after* the event that triggered neural activity: The sluggishness of this response is one of the major drawbacks of functional neuroimaging, as it must be accounted for when designing an experiment and analyzing the data.

After collection, fMRI images need to be preprocessed to remove various sources of noise. A typical preprocessing procedure (e.g., Friston et al., 2006) consists of the following steps:

1. *Spatial realignment*, whereby head motion from one image to another is removed by means of rigid-body transformation;

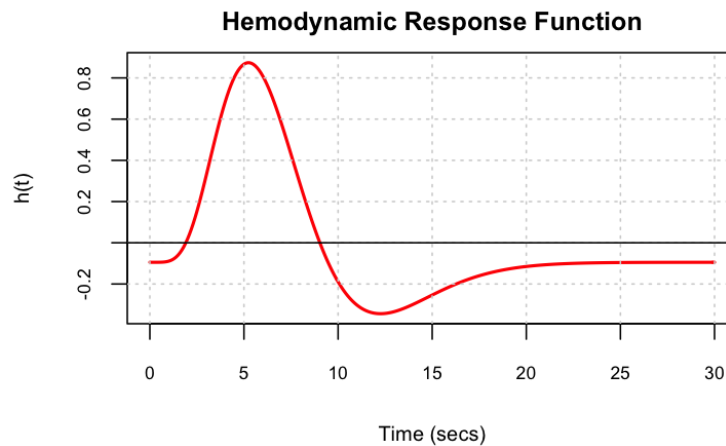


Figure 1: The hemodynamic response function, as implemented in the `fmri` package.

2. *Slice timing correction*, whereby the temporal differences in the time at which voxels belonging to different 2D slices are removed;
3. *Normalization*, where individual brains are warped onto a common anatomical template by means of a non-linear transformation. This step is crucial to aggregate and compare data collected from anatomically different brains;
4. *Spatial smoothing*, which is typically performed with a 3D Gaussian kernel and is aimed at reducing noise and removing residual differences in individual anatomy between participants.

Specialized R packages exist that perform the first and the last of these steps. Specifically, package `RNiftyReg` (Clayden, 2013) provides functions for realigning images in the NIFTI file format, while spatial smoothing is available in the `AnalyzefMRI` package (Bordier et al., 2011). In addition, the `fmri` package implements an advanced smoothing procedure that adapts to the constraints of the underlying anatomical structures.

To the best of my knowledge, and as confirmed by Tabelow and Polzehl (2011), there are currently no R packages that support the second and third preprocessing steps, which must still be performed using other software.

Statistical analysis of fMRI data

fMRI data is typically analyzed with a *mass-univariate* approach (Friston et al., 2006), where a single model is fitted to the timecourse of hemodynamic activity of each voxel independently. This is done by first creating a $n \times m$ matrix of regressors \mathbf{X} , where the m columns represent the time courses of different experimental conditions and the n rows represent the discrete time points at which the images were acquired. To account for the sluggishness of the BOLD response (Figure 1), the matrix \mathbf{X} is created by convolving each column of a matrix \mathbf{S} describing onsets and durations of the experimental stimuli with the hemodynamic response function h , so that $\mathbf{X} = h * \mathbf{S}$.

Data from all the voxels \mathbf{Y} of a single participant is then analyzed by fitting the linear model $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, and thus finding the values of $\boldsymbol{\beta}$ that minimize the residual sum of squares $\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$ for each voxel. However, the experimenter is typically interested in examining *differences* between conditions, rather than the estimates of the conditions *per se*. These differences are often referred to as “contrasts”, and can be expressed as the product between a contrast vector \mathbf{c} (usually a binary vector), and the vector of estimates $\boldsymbol{\beta}$, i.e., $\mathbf{c}^T \boldsymbol{\beta}$. Thus, the null hypothesis H_0 that there is no difference between conditions corresponds to the case $\mathbf{c}^T \boldsymbol{\beta} = 0$.

The result of a neuroimaging analysis is a set of three-dimensional images. Each image is made of tens of thousands of *voxels*, and each voxel contains the statistical value of a test of a contrast vector \mathbf{c} . These parametric images are typically thresholded at a level that corresponds to a significant value of the corresponding statistical test, corrected for multiple comparisons (Friston et al., 2006). This procedure yields a set of 3D *clusters* of spatially adjacent voxels, whose activity is similarly affected by the experimental manipulations.

While R does not provide many tools for preprocessing, there are many R packages that can be used to analyze preprocessed fMRI datasets. For instance, the `fmri` package contains functions to fit statistical models to fMRI datasets in the way described above. In particular, the design matrix \mathbf{X} is generated using the function `fmri.design`, and a linear model can be fit using `fmri.lm`. In

addition, both the **AnalyzefMRI** and the **fmri** packages also contain functions to perform Independent Component Analysis in both the spatial and temporal domains, as well as several procedures for multiple-error correction, such as Bonferroni, False Discovery Rate, and various instances of family-wise corrections. The package **RfmriVC** (Bothmann and Kalus, 2013) permits users to modulate the effects of the hemodynamic response function with other variables that vary within a single session, such as time or fatigue. The package **arf3DS4** (Weeda et al., 2011) uses the estimated β - or t -images to identify active brain regions and their patterns of functional connectivity. The package **BHMSMAfMRI** (Sanyal and Ferreira, 2014) implements Bayesian multi-subject analysis. Finally, the package **neuRosim** (Welvaert et al., 2011) contains functions that simulate brain imaging data for testing purposes. A comprehensive list of these and related packages is maintained in the CRAN task view on Medical Image Analysis (<http://CRAN.R-project.org/view=MedicalImaging>).

Describing the functionalities of these packages is beyond the scope of this paper; the interested reader can check the recent and extensive review by Eloyan et al. (2014) on the subject. Instead, the remainder of this article will introduce some simple code lines that demonstrate how to read imaging data, plot them over a brain image, and visually compare results from different neuroimaging studies. Incidentally, this type of comparison is a common way to familiarize oneself with a new field in neuroimaging research, and is quite often an important step in justifying hypotheses about the functions of one brain region.

An example of coordinate-based meta-analysis

To understand the coordinate-based meta-analysis, we need to take a step back. Remember that the result of an fMRI analysis is a set of clusters of voxels that are jointly above a certain statistical threshold. Because a single test might identify multiple clusters, neuroimaging papers typically focus only on those that are deemed the most interesting, and relegate the full list of clusters to separate tables. These tables detail various characteristics of each cluster, including its size (i.e., the number of voxels it spans) and position within the brain.

To facilitate comparisons across studies, cluster positions are given in terms of three-dimensional coordinates in predefined stereotactic spaces (which will be discussed in the next section). Because a cluster spans many voxels, different conventions can be applied to determine which coordinates best represent a cluster's position. For instance, some authors might report the coordinates of the cluster's "peak" (i.e., the highest parameter value), while others report the cluster's geometrical center or its center of mass. Fortunately, these values tend not to differ much from each other, and for simplicity we can consider them as functionally equivalent.

In theory, and assuming they are available and comparable, one could directly enter contrast or statistical parameter images from different studies into a meta-analysis. This procedure is known as *image-based* meta-analysis, and is generally considered the most reliable (Salimi-Khorshidi et al., 2009). However, because cluster sizes are dependent on the statistical error correction procedures that were adopted, most meta-analyses of fMRI data, like the Activation Likelihood Estimate (Turkeltaub et al., 2002), focus instead on the consistency of cluster *positions*, i.e., how often a given region is reported, and how the spatial distribution of cluster peaks is affected by the study parameters. This procedure is known as *coordinate-based* meta-analysis, and has been successfully used to identify possible subdivisions within the same anatomical brain regions (for example, the rostral prefrontal cortex, Burgess et al. 2007; or the anterior cingulate cortex, Bush et al. 2000), or to suggest the involvement, in a certain cognitive function, of a brain structure that had been previously overlooked (e.g., the basal ganglia in language switching, Stocco et al. 2014). The remainder of this paper will introduce the packages and the R code that will be used to visualize foci of activation on a standard brain template. The final result will be an image like the one in Figure 2.

Although simple, the code provided herein is useful in its own right. Images like Figure 2 have been published in many scientific papers, such as Bush et al. (2000) and Burgess et al. (2007). This very same code, in fact, has been used to generate two figures published in Stocco et al. (2014) and Buchweitz and Prat (2013).

Displaying the MNI template

The spatial coordinates of a cluster of voxels are given in reference to an *ideal brain template*, oriented in a given *stereotactic space*. Two stereotactic spaces are commonly used, the Talairach-Tournoux and the Montreal Neurological Institute (MNI) spaces. In this paper, we will assume that all the coordinates are in the MNI space, which is preferred by the majority of researchers (Poldrack et al., 2011). Talairach-Tournoux coordinates can be converted to their MNI equivalent using non-linear transformations, such as those proposed by Brett et al. (2001).

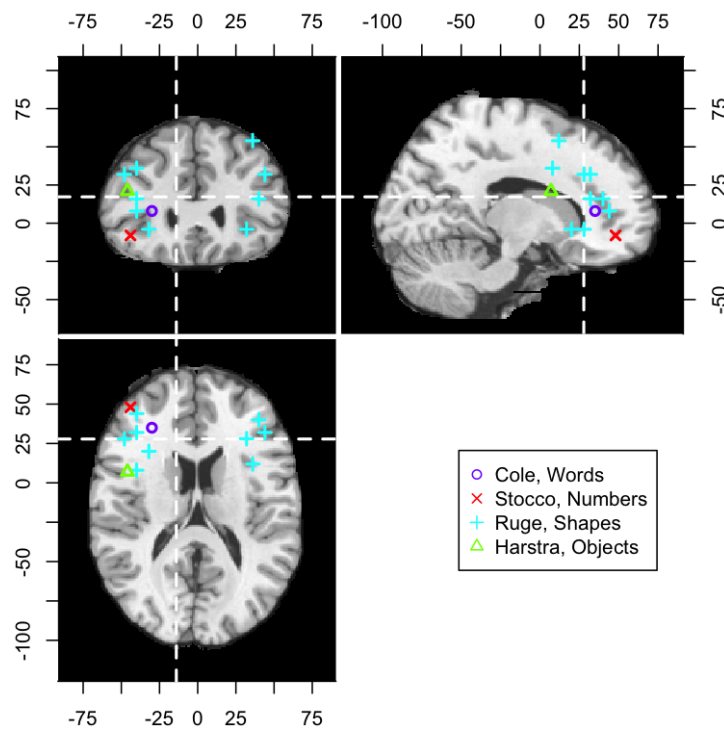


Figure 2: A visual summary of the results of four neuroimaging studies.

In the MNI space the x , y , and z axes correspond to the left-right, front-back, and up-down orientation of a brain. The origin of the space is located in a point in the middle of the brain, so that the brain volume spans both positive and negative coordinates along all axes.

The first step to producing Figure 2 is to visualize the MNI template brain on an R plot. Many versions of the MNI template exist; in this paper, I am going to use the so-called "Colin27" version (Holmes et al., 1998). This template is available on various websites, and is included in many common software packages for fMRI, such as MRICron (Rorden et al., 2000). This example will use the template that is available at http://packages.bic.mni.mcgill.ca/mni-models/colin27/mni_colin27_1998_nifti.zip.

The template is encoded in a particular file format known as NIFTI, which is, by far, the most common file format for functional neuroimages. All the packages mentioned above contain functions to read and write NIFTI files. In this example, I will use the `oro.nifti` package (Whitcher et al., 2011), which has been developed specifically for this purpose. Assuming that the template's URL is stored in the variable `colin.url`, the image can be easily loaded with this code:

```
library(oro.nifti)
temp <- file.path(tempdir(), "mni_colin27_1998_nifti.zip")
colin.url <-
  "http://packages.bic.mni.mcgill.ca/mni-models/colin27/mni_colin27_1998_nifti.zip"
download.file(colin.url, dest = temp)
unzip(temp, exdir = tempdir())
colin <- readNIFTI(file.path(tempdir(), "colin27_t1_tal_lin.nii"))
```

The `colin` object is simply a 3D matrix. The matrix dimensions are of $181 \times 217 \times 181$ voxels, and can be accessed by calling the `dim` function: `dim(colin)`. Each slice of this matrix is in itself a 2D image, and can be visualized with any standard 2D plotting tools in R. For instance, a professional-looking sagittal (i.e., lateral) view of the brain can be visualized by calling the `image` function on the 80th 2D slice across the x axis, e.g., `image(colin[80, ,], col = grey(0:255 / 255))`, as shown in Figure 3.

Notice that the original template, as shown in Figure 3, contains the brain as well as a surrounding skull. For clarity purposes, the skull has been omitted in Figure 2. The Colin27 archive that had been downloaded in the previous step also contains a "brain mask", i.e., a binary image that spans the entire brain but leaves out the skull. An skull-stripped image of the brain can be generated as the point-wise product of the `colin` image and the mask:

```
mask <- readNIFTI(file.path(tempdir(), "colin27_t1_tal_lin_mask.nii"))
colin <- colin * mask
```

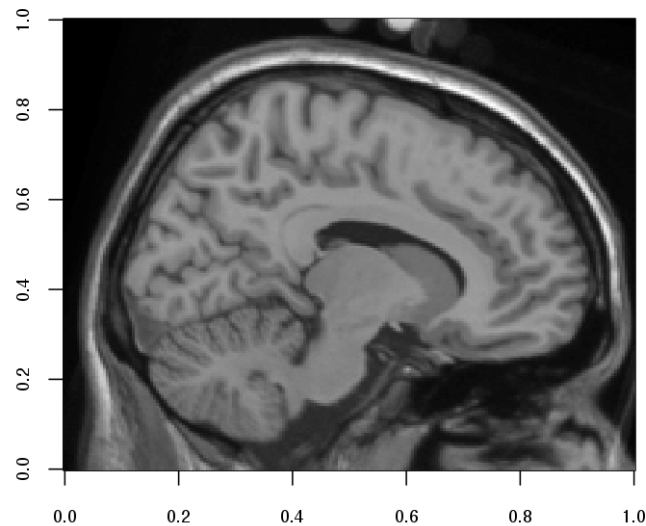



Figure 3: The 80th 2D sagittal slice of the colin template.

The three axes of the colin object are already aligned with the three axes of the colin template. However, the coordinate system of the image and the coordinate system of the colin image have different centers. In the colin object, the point of coordinates $x = 0$, $y = 0$, $z = 0$ is located at the leftmost voxel of the first row of the bottom slice. In the MNI space, however, the origin is located in the middle of the brain. The MNI origin corresponds to the voxel of coordinates $x = 91$, $y = 126$, $z = 72$ in the image space. Thus, the MNI coordinates need to be translated in space to be visualized correctly. Because each voxel in colin is a $1 \times 1 \times 1$ mm cube, this affine translation can be easily implemented using sweep:

```
# Origin of MNI space
origin <- c(x = 91, y = 126, z = 72)

# Converting back and forth to MNI space
mni2xyz <- function(x) sweep(x, 2, as.double(origin), "+")
xyz2mni <- function(x) sweep(x, 2, as.double(origin), "-")
```

Figure 2 is a rendition of the brain in so-called *orthographic view*, which is particularly helpful for visualizing the relative position of foci within the brain. The orthographic view consists of three orthogonal brain slices that intersect in a specific point. We will refer to this point as the center C of the orthographic view, with coordinates x_C , y_C , z_C . The three views are produced by simply selecting the 2D slice of the 3D image with the corresponding values of C . The top left view in Figure 2 is called *coronal view*, and corresponds to the case where $y = y_C$. The top right view in Figure 2 is called *sagittal view*, and corresponds to the case where $x = x_C$. Finally, the bottom-left view is known as *axial view*, and corresponds to the case where $z = z_C$.

The package `oro.nifti` has its own method (`orthographic`) to visualize a brain in orthographic view. In this example, however, we will implement the procedure from scratch, as it is fairly easy and allows for more control on the proper visualization.

The first step towards producing the image in Figure 2 is to create a the plot window and divide it into four quadrants. R's default layout method is a convenient way to do so:

```
layout(matrix(1:4, ncol = 2, byrow = TRUE),
       heights = c(181, 217),
       widths = c(181, 217))
layout.show(4)
```

The heights and widths parameters in the code are used to assign non-uniform heights and widths to the four quadrants. This is needed, in turn, because the colin template is not cubic, and it spans 181 voxels along the x and z axis, but 217 on the y axis.

Next, we need to define our center point C . Initially, I will simply assume that the center of the orthographic view is also the origin in MNI space, i.e., the point of coordinates $x = 0$, $y = 0$, $z = 0$. The coronal, sagittal, and axial views of the orthographic figure can be simply generated by subsequent

calls of the default method `image`:

```
center <- origin
greys <- grey(1:max(colin) / max(colin))
par(mar = c(2, 2, 2, 2))
image(colin[, center["x"], ], col = greys) # Coronal
image(colin[center["y"], , ], col = greys) # Sagittal
image(colin[, , center["z"]], col = greys) # Axial
plot.new() # Empty quadrant
```

In the code above, the `greys` variable is used to create as many shades of grey as there are values of intensity in the `Colin27` template. These colors are passed as an argument to visualize the brain slices in the canonical greyscale (instead of R's default heat colors) like in Figure 3. The `mar` variable sets the plot margins for each image to 2 lines of text, which is the minimum amount necessary to visualize the image axes in each view. Finally, the call to `plot.new` is required because R expects one plot for each quadrant defined by layout.

Plotting the foci

Because each image in the orthographic view is a figure in itself, the individual foci can be visualized by plotting points on top of each image with R's `points` function, before proceeding to the next image. In the example of Figure 2, the individual foci of activation were stored in a 'data.frame' object called `foci`.

```
> foci
   x  y  z Study  Type
1 -30 35 8  Cole  Words
2 -46 7 21 Harstra Objects
3 -40 8 36  Ruge  Shapes
4 36 12 54  Ruge  Shapes
5 -48 28 32  Ruge  Shapes
6 44 32 32  Ruge  Shapes
7 -40 32 16  Ruge  Shapes
8 -40 44 8  Ruge  Shapes
9 40 40 16  Ruge  Shapes
10 -32 20 -4  Ruge  Shapes
11 32 28 -4  Ruge  Shapes
12 -44 48 -8 Stocco Numbers
```

This list of foci was created by examining the results of four publications that, between 2010 and 2012, defined a rule-learning paradigm called Rapid Instructed Task Learning (Cole et al., 2013), which can be used to study how the brain encodes abstract rules. Specifically, the foci of activation in this list come from an analysis of the brain regions that were significantly more engaged when encoding completely *novel* rules than rules that had been practiced before. For simplicity's sake, only foci appearing in the frontal lobe were considered. Interestingly, these four studies used rules of similar length and complexity, but of different nature. Specifically, the rules used in the four studies applied to pictures of common objects (Hartstra et al., 2011), geometric shapes (Ruge and Wolfensteller, 2010), words (Cole et al., 2010), and numbers (Stocco et al., 2012). By comparing the distribution of results across these studies, we can observe whether the *nature* of the rules performed determines the *location* where the corresponding rule is represented. For example, the distribution of foci in Figure 2 seems to suggest that rules that encode more abstract materials (such as words and numbers) activate more anterior regions than rules that encode more concrete materials (such as pictures of common objects).

In the `foci` data frame, the `x`, `y`, and `z` columns represent, of course, the corresponding MNI coordinates of each focus of activation. The fourth column `Study` serves to identify the specific study the foci belong to. The last column `Type` defines the type of stimuli the rules were applied to in each study.

Before plotting the foci, their coordinates need to be transformed from the three-dimensional MNI space to the bidimensional space of each image. The procedure requires two consecutive steps. First, the coordinates need to be transformed from MNI space to image space (using the `mni2xyz` function above). Second, the image coordinates need to be properly scaled to the fit within the range `[0, 1]` of the axes generated by the `image` function. Once we have the foci coordinates in the proper space, we can plot them by using the appropriate coordinates as the values of the `x` and `y` parameters of `points`. The following code illustrates the example.

```
# Transforms the coordinates
im.foci <- mni2xyz(foci[1:3])
```

```
# Normalizes
im.foci <- sweep(im.foci, 2, dim(colin), "/")

image(colin[, center["y"], ], col = greys) # Coronal
points(x = im.foci[, "x"], y = im.foci[, "z"], lwd = 2, col = "red")

image(colin[center["x"], , ], col = greys) # Sagittal
points(x = im.foci[, "y"], y = im.foci[, "z"], lwd = 2, col = "red")

image(colin[, , center["z"]], col = greys) # Axial
points(x = im.foci[, "x"], y = im.foci[, "y"], lwd = 2, col = "red")
plot.new()
```

This code, however, does not efficiently convey the position of the foci. This is because, while the three views are centered at the origin of the MNI coordinates, the foci are instead mostly distributed in the left hemisphere and in the frontal lobe. The easiest way to solve this problem is to use the center C_F of the distribution of foci, and not the MNI origin, as the center of the orthographic view. The coordinates of C_F are simply the mean of each focus' x , y , and z coordinates: $x_{C_F} = \bar{x}$, $y_{C_F} = \bar{y}$, $z_{C_F} = \bar{z}$. In R, these values can be easily calculated with one call of the `colMeans` function, and turned into integers by using `round`.

```
im.foci <- mni2xyz(foci[1:3])
center <- round(colMeans(im.foci), 0)
```

While this produces a more meaningful figure, it does not resolve all the problems. For instance, the sagittal view of Figure 2 could come from either the left or the right hemisphere. Because the left and right hemispheres are symmetric, there is no way to know from which one a sagittal view comes from. To avoid this ambiguity, orthographic views often outline the two planes that are perpendicular to each view in the form of lines. Thus, each of the three images shows the position of the planes that define the other two images. Because the three images are orthogonal with each other, each plane is visualized as a vertical or a horizontal line (dotted white lines in Figure 2). These lines can be drawn by calling `abline` before invoking the `points` function, and using each plane's coordinates as the `h` or `v` parameter of `abline`. The plane coordinates are, of course, those of the foci's center C_F , normalized to be in the range $[0, 1]$.

```
# Normalizes
im.foci <- sweep(im.foci, 2, dim(colin), "/")
CF <- colMeans(im.foci)

image(colin[, center["y"], ], col = greys) # Coronal
abline(v = CF["x"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[, "x"], y = im.foci[, "z"], lwd = 2, col = "red")

image(colin[center["x"], , ], col = greys) # Sagittal
abline(v = CF["y"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[, "y"], y = im.foci[, "z"], lwd = 2, col = "red")

image(colin[, , center["z"]], col = greys) # Axial
abline(v = CF["x"], h = CF["y"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[, "x"], y = im.foci[, "y"], lwd = 2, col = "red")

plot.new() # Empty quadrant
```

One important feature of Figure 2 is the use of tick marks and labels representing MNI coordinates, instead of the default $[0, 1]$ axes that are generated otherwise. In addition, the tick marks and labels of Figure 2 are positioned on the outside of the plotting region, bringing the three views closer together. Both of these features increment the readability of the plot.

The precise position of tick marks is controlled by the `at` parameter of the `axis` function, which specifies the location of the tick marks in the axis' native range (in this case, $[0, 1]$). The corresponding labels are similarly controlled by the `labels` parameter.

Let us start by defining a set of "useful" position marks in MNI space, e.g., every 25mm from -100 to 100 , along the x , y , and z axis. These labels can then be transformed in 2D plotting coordinates by first converting them into image space and subsequently normalizing them.

```
labels <- seq(-100, 100, by = 25)
ticks <- cbind(x = labels, y = labels, z = labels)
```

```
ticks <- sweep(mni2xyz(ticks), 2, dim(colin), "/")
```

Now that we have a data frame of tick marks, we can proceed to modify the position of the axes in each image of the orthographic view. To do this, we first need to suppress the image method's default axes, which can be done by setting the parameters `ann` (for the labels) and `axes` (for the axes) to `FALSE`. Second, we need to create custom axes with the `axis` function. The axis position is the first argument of the `axis` function, and it is indexed by a number between 1 and 4, in clockwise order beginning from the bottom.

Keeping this scheme in mind, we want to plot axes 1 and 3 on the coronal view; axes 3 and 4 on the sagittal view; and axes 1 and 2 on the axial view. We also want to modify the margins (using `par(mar = ...)`) of each view, so that the outside margins are larger than the inside margins to make room for axes, tick marks, and labels. The resulting code is this:

```
par(mar = c(0.1, 2, 2, 0.1)) # Coronal
image(colin[, center["y"], ], ann = FALSE, axes = FALSE, col = greys)
axis(2, pos = 0, at = ticks["z"], labels = labels)
axis(3, pos = 1, at = ticks["x"], labels = labels)
abline(v = CF["x"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci["x"], y = im.foci["z"], lwd = 2, col = "red")

par(mar = c(0.1, 0.1, 2, 2)) # Sagittal
image(colin[center["x"], , ], ann = FALSE, axes = FALSE, col = greys)
axis(3, pos = 1, at = ticks["y"], labels = labels)
axis(4, pos = 1, at = ticks["z"], labels = labels)
abline(v = CF["y"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci["y"], y = im.foci["z"], lwd = 2, col = "red")

par(mar = c(2, 2, 0.1, 0.1)) # Axial
image(colin[, , center["z"]], ann = FALSE, axes = FALSE, col = greys)
axis(1, pos = 0, at = ticks["x"], labels = labels)
axis(2, pos = 0, at = ticks["y"], labels = labels)

abline(v = CF["x"], h = CF["y"], lty = 2, lwd = 2, col = "white")
points(x = im.foci["x"], y = im.foci["y"], lwd = 2, col = "red")

# Last quadrant
plot.new()
```

Because neuroimaging meta-analysis often aims at identifying functional specializations within or between brain regions, it is helpful to color-code the given foci of activation according to some category. In the example of Figure 2, the foci of activation come from four different studies that dealt with learning rules for different types of stimuli.

The study and the stimulus type are encoded in the fourth and fifth column of the `foci` data frame, and are indicated in Figure 2 by the color and point symbol of each focus of activation. Thus, the number of colors and the number of point marks are determined by the number of unique values of the variable `foci$Study` and the variable `foci$Type`. To visualize each point with the proper marker and color, we need to create a color vector and a point mark vector, each of which is as long as the set of foci. To cover all the possible combinations, the color and mark vectors need to be created by substituting each value of the `Rule` and `Learning` columns in the `foci` data frame with the corresponding color and mark.

```
foci.marks <- as.integer(foci$Learning)
foci.colors <- rainbow(nlevels(foci$Rule))[as.integer(foci$Rule)]
```

Now we can just modify the `points` function used above by specifying the proper parameters—`col` for the colors and `pch` for the marks.

```
# Coronal view code
...
points(x = im.foci["x"], y = im.foci["z"], lwd = 2, pch = foci.marks,
       col = foci.colors)

# Sagittal view code
...
points(x = im.foci["y"], y = im.foci["z"], lwd = 2, pch = foci.marks,
       col = foci.colors)
```

```
# Axial view code
...
points(x = im.foci[, "x"], y = im.foci[, "y"], lwd = 2, pch = foci.marks,
       col = foci.colors)
```

The last step to complete Figure 2 is to add the figure legend. Because our color and point mark vectors contain combinations of color and point mark, we need to simplify them appropriately, so that we have only one combination of each. To do so, we will first create a data frame object holding all the legend arguments, and then use `unique` to extract the unique combinations of values. Thus, the final piece of the code will contain the following lines:

```
# Last quadrant
plot.new()
l.args <- data.frame(col = foci.colors, pch = foci.marks,
                    stringsAsFactors = FALSE)
l.args$legend <- paste(foci$Learning, foci$Rule)
l.args <- unique(l.args)
legend(x = "center", legend = l.args$legend, col = l.args$col, pch = l.args$pch)
```

Summary

This paper has discussed the application of R to functional neuroimaging. The first part of this article has given an overview of the basics of functional neuroimaging data acquisition, preprocessing, and analysis. Neuroimaging data requires specific preprocessing steps, such as spatial realignment and normalization, that are currently not covered by R packages. The analysis and modeling of neuroimaging data, on the other hand, is covered by many specialized packages. So, why should one use R instead of, let's say, AfNI or FSL? First and foremost, the existing R packages provide unique and sophisticated features that are not available otherwise. Second, R by itself provides hundreds of statistical packages and models that are *not* specific to neuroimaging, but that can be readily applied to supplement, expand, and innovate the existing approaches. For example, one can imagine to use the `spatstat` package (Baddeley and Turner, 2005) to analyze the three-dimensional spatial distribution of foci in Figure 2, thus giving some statistical bases for our intuition that different types of stimuli affect how rules are represented.

As an introduction on how R can be used to handle neuroimaging data, the second part of this paper has presented an example of R code that visualizes the results of different neuroimaging studies on an orthographic view of the brain. The code permits to visually inspect the distribution of reported cluster of activation across studies, which is an important first step for generating more precise hypotheses about the functions of different brain regions. It uses brain templates and coordinate spaces that are standard in the neuroimaging literature, and can generate journal-quality images for publication. In fact, this very same code has been used to generate figures in published reviews (Stocco et al., 2014; Buchweitz and Prat, 2013).

In summary, although R cannot completely replace specialized software for fMRI data analysis, it can readily integrate them—which is relatively easy after familiarizing oneself with the formats, templates, and conventions of neuroimaging research.

Bibliography

- A. Baddeley and R. Turner. `spatstat`: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. URL <http://www.jstatsoft.org/v12/i06>. [p13]
- C. Bordier, M. Dojat, and P. L. de Micheaux. Temporal and spatial independent component analysis for fMRI data sets embedded in the `AnalyzefMRI` R package. *Journal of Statistical Software*, 44(9): 1–24, 2011. URL <http://www.jstatsoft.org/v44/i09/>. [p6]
- L. Bothmann and S. Kalus. *RfmriVC: Varying stimulus coefficient fMRI models in R*, 2013. URL <http://CRAN.R-project.org/package=RfmriVC>. R package version 1.0.4. [p7]
- M. Brett, K. Christoff, R. Cusack, and J. Lancaster. Using the Talairach atlas with the MNI template. *Neuroimage*, 5(13):85, 2001. [p7]
- A. Buchweitz and C. Prat. The bilingual brain: Flexibility and control in the human cortex. *Physics of Life Reviews*, 10(4):428–443, 2013. [p7, 13]

- P. W. Burgess, S. J. Gilbert, I. Dumontheil, P. W. Burgess, S. J. Gilbert, and I. Dumontheil. Function and localization within rostral prefrontal cortex (area 10). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):887–899, 2007. [p7]
- G. Bush, P. Luu, and M. I. Posner. Cognitive and emotional influences in anterior cingulate cortex. *Trends in Cognitive Sciences*, 4(6):215–222, 2000. [p7]
- J. Clayden. *RNiftyReg: Medical image registration using the NiftyReg library*, 2013. URL <http://CRAN.R-project.org/package=RNiftyReg>. R package version 1.1.2; based on original code by Marc Modat and Pankaj Daga. [p6]
- M. W. Cole, A. Bagic, R. Kass, and W. Schneider. Prefrontal dynamics underlying rapid instructed task learning reverse with practice. *The Journal of Neuroscience*, 30(42):14245–14254, 2010. [p10]
- M. W. Cole, P. Laurent, and A. Stocco. Rapid instructed task learning: A new window into the human brain’s unique capacity for flexible cognitive control. *Cognitive, Affective, & Behavioral Neuroscience*, 13(1):1–22, 2013. [p10]
- R. W. Cox and J. S. Hyde. Software tools for analysis and visualization of fMRI data. *NMR in Biomedicine*, 10(45):171–178, 1997. [p5]
- A. Eloyan, S. Li, J. Muschelli, J. J. Pekar, S. H. Mostofsky, and B. S. Caffo. Analytic programming with fMRI data: A quick-start guide for statisticians using R. *PLoS ONE*, 9(2):e89470, 2014. [p7]
- K. Friston, J. Ashburner, S. Kiebel, T. Nichols, and W. Penny, editors. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press, 1 edition, Dec. 2006. ISBN 0123725607. URL <http://www.sciencedirect.com/science/book/9780123725608>. [p5, 6]
- E. Hartstra, S. Kühn, T. Verguts, and M. Brass. The implementation of verbal instructions: An fMRI study. *Human Brain Mapping*, 32(11):1811–1824, 2011. [p10]
- C. J. Holmes, R. Hoge, L. Collins, R. Woods, A. W. Toga, and A. C. Evans. Enhancement of MR images using registration for signal averaging. *Journal of Computer Assisted Tomography*, 22(2):324–333, 1998. [p8]
- R. A. Poldrack, J. A. Mumford, and T. E. Nichols. *Handbook of Functional MRI Data Analysis*. Cambridge University Press, 2011. [p7]
- J. Polzehl and K. Tabelow. fmri: A package for analyzing fmri data. *R News*, 7(2):13–17, 2007. [p5]
- C. Rorden, M. Brett, et al. Stereotaxic display of brain lesions. *Behavioural Neurology*, 12(4):191–200, 2000. [p8]
- H. Ruge and U. Wolfensteller. Rapid formation of pragmatic rule representations in the human brain during instruction-based learning. *Cerebral Cortex*, 20(7):1656–1667, 2010. [p10]
- G. Salimi-Khorshidi, S. M. Smith, J. R. Keltner, T. D. Wager, and T. E. Nichols. Meta-analysis of neuroimaging data: A comparison of image-based and coordinate-based pooling of studies. *Neuroimage*, 45(3):810–823, 2009. [p7]
- N. Sanyal and M. A. Ferreira. *BHMSMAfMRI: Bayesian hierarchical multi-subject multiscale analysis of functional MRI data*, 2014. URL <http://CRAN.R-project.org/package=BHMSMAfMRI>. R package version 1.0. [p7]
- S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobnjak, D. E. Flitney, et al. Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage*, 23:S208–S219, 2004. [p5]
- A. Stocco, C. Lebiere, R. C. O’Reilly, and J. R. Anderson. Distinct contributions of the caudate nucleus, rostral prefrontal cortex, and parietal cortex to the execution of instructed tasks. *Cognitive, Affective, & Behavioral Neuroscience*, 12(4):611–628, 2012. [p10]
- A. Stocco, B. Yamasaki, R. Natalenko, and C. S. Prat. Bilingual brain training: A neurobiological framework of how bilingual experience improves executive function. *International Journal of Bilingualism*, 18(1):67–92, 2014. [p7, 13]
- K. Tabelow and J. Polzehl. Statistical parametric maps for functional MRI experiments in R: The package fmri. *Journal of Statistical Software*, 44(11):1–21, 2011. URL <http://www.jstatsoft.org/v44/i11/>. [p5, 6]

- P. E. Turkeltaub, G. F. Eden, K. M. Jones, and T. A. Zeffiro. Meta-analysis of the functional neuroanatomy of single-word reading: Method and validation. *Neuroimage*, 16(3):765–780, 2002. [p7]
- W. D. Weeda, F. de Vos, L. J. Waldorp, R. P. P. Grasman, and H. M. Huizenga. arf3DS4: An integrated framework for localization and connectivity analysis of fMRI data. *Journal of Statistical Software*, 44(14):1–33, 2011. URL <http://www.jstatsoft.org/v44/i14/>. [p7]
- M. Welvaert, J. Durnez, B. Moerkerke, G. Verdoolaege, and Y. Rosseel. neuRosim: An R package for generating fMRI data. *Journal of Statistical Software*, 44(10):1–18, 2011. URL <http://www.jstatsoft.org/v44/i10/>. [p7]
- B. Whitcher, V. J. Schmid, and A. Thornton. Working with the DICOM and NIfTI data standards in R. *Journal of Statistical Software*, 44(6):1–28, 2011. URL <http://www.jstatsoft.org/v44/i06/>. [p8]
- K. J. Worsley, C. Liao, J. Aston, V. Petre, G. Duncan, F. Morales, and A. Evans. A general statistical analysis for fMRI data. *Neuroimage*, 15(1):1–15, 2002. [p5]

Andrea Stocco
Department of Psychology and Institute for Learning and Brain Sciences
University of Washington
Campus Box 357988, Seattle, WA
USA
stocco@uw.edu

Automatic Conversion of Tables to LongForm Dataframes

by Jimmy Oh

Abstract `TableToLongForm` automatically converts hierarchical Tables intended for a human reader into a simple LongForm dataframe that is machine readable, making it easier to access and use the data for analysis. It does this by recognising positional cues present in the hierarchical Table (which would normally be interpreted visually by the human brain) to decompose, then reconstruct the data into a LongForm dataframe. The article motivates the benefit of such a conversion with an example Table, followed by a short user manual, which includes a comparison between the simple one argument call to `TableToLongForm`, with code for an equivalent manual conversion. The article then explores the types of Tables the package can convert by providing a gallery of all recognised patterns. It finishes with a discussion of available diagnostic methods and future work.

Introduction

`TableToLongForm` is an R package that automatically converts hierarchical Tables¹ intended for a human reader into a LongForm R "data.frame" that is machine readable.² While several tools exist that can aid in manipulation of data, such as OpenRefine (OpenRefine, 2013), which can be used to clean messy data, the `speedr` R package (Visne et al., 2012), which can aid in filtering data, and R packages like `reshape2` (Wickham, 2007) and `plyr` (Wickham, 2011), which enable restructuring of the data to focus on specific aspects of the data, for these tools to work their magic you first need machine readable data. However, data released as Tables are *not* machine readable. At best, such tools will provide some aid in manually converting the data to something that is machine readable, a process that is costly in terms of both time and effort. `TableToLongForm` is an automatic tool for converting a family of Tables to a machine readable form, and once so converted the user is free to use their favourite tool, R or otherwise, to make full use of the data.

The article motivates the benefit of such a conversion with an example Table, followed by a short user manual, which includes a comparison between the simple one argument call to `TableToLongForm`, with code for an equivalent manual conversion. The article then explores the types of Tables the package can convert by providing a gallery of all recognised patterns. It finishes with a discussion of available diagnostic methods and future work.

Motivation

There is still a prevalence of data releases being made for direct human consumption in formats that are not machine readable, a significant barrier to effective utilisation of the data. One symptom of this is the release of data in tabular form that relies on a hierarchy that can only be understood after identifying patterns and discerning the structure of the Table, a task easy for a human brain but rather difficult for a computer.

An example of such a Table is shown in Figure 1. For such a Table, the computer will be unable to easily read in the data due to the difficulty in finding all information related to a piece of data. Take the number '876' in cell (5, 9) for instance; to collect all the information linked to that number we must look at cell (5, 1) for the time period ('2007Q4'), cell (4, 9) for the data heading ('Total Labour Force'), cell (3, 2) for the ethnic category ('European Only') and cell (2, 2) for the gender category ('Male'). Note that, aside from the time period and the data heading, the other information related to cell (5, 9) were neither in the same row nor the same column. The human brain can interpret the positional cues to understand the hierarchy fairly easily; the computer requires a lot more work.

Preparing such data for use would normally require a non-trivial time investment to restructure the data in a manner that can be machine read and used. If such preparatory work was done manually, such work will have to be repeated multiple times as the data is updated. In some cases the data will be spread across multiple files, which means that much more preparatory work. Even if the work is scripted, small changes in the format can easily throw a wrench into the works and break it. All of this quickly adds up to a significant time cost to make use of data released in Tables.

¹Table, with a capital T, is used in this article to specifically mean hierarchical tables, e.g. Figure 1.

²Machine readable is used to mean that the format is intended for access and manipulation by computers, and it is thus much easier to use the data for various purposes, such as statistical analysis. It can alternatively be described as *Tidy Data* (Wickham, 2014), with the conversion taking the data closer to the ideal 'tidy' form.

LongForm is a simple alternative data format that most R users will find familiar as an R "data.frame" object, the format that most R functions require of their data. **TableToLongForm** automatically converts Tables to LongForm dataframes,³ which can mean significant savings in time while enabling much greater utilisation of the data. Figure 2 shows Figure 1 after automatic conversion using **TableToLongForm**, which took around a tenth of a second on the author's machine. In particular, note that the same data we examined above, '876' now in cell (2, 11), has all related information in the same row (except for the column heading, which is in the same column), making it easy for the computer to understand and manipulate.

	1	2	3	4	5	6	7	8	9	10	11
1	Labour Force Status by Sex by Sing/Comb Ethnic Group (Qrtly--Mar/Jun/Sep/Dec)										
2		Male									
3		European Only								Maori Only	
4		Persons Em	Persons Un	Not in Labo	Working Ag	Labour Forc	Unemployr	Employer	Total Labou	Persons Em	Persons Un
5	2007Q4	856	20	280	1,156	76	2	74	876	71	6
6	2008Q1	863	25	284	1,172	76	3	74	888	69	8
7	2008Q2	850	26	281	1,157	76	3	74	876	67	6
8	2008Q3	840	30	286	1,155	75	3	73	869	72	9
9	2008Q4	855	30	275	1,159	76	3	74	884	76	8
10	2009Q1	845	35	279	1,160	76	4	73	880	75	8
11	2009Q2	832	35	280	1,146	76	4	73	866	74	10
12	2009Q3	813	42	290	1,146	75	5	71	856	71	11
13	2009Q4	831	40	277	1,148	76	5	72	871	72	14
14	2010Q1	822	36	283	1,142	75	4	72	859	72	11

Figure 1: An example of a hierarchical Table. The Table is of the Labour Force Status data (Statistics New Zealand, 2013) and in total spans 240 columns. The Table is too large to be immediately useful for humans, and cannot even be easily manipulated with a computer, as understanding the data requires linking information across different rows and columns.

	1	2	3	4	5	6	7	8	9	10	11
1				Persons Em	Persons Un	Not in Labo	Working Ag	Labour Forc	Unemployr	Employer	Total Labou
2	Male	European C	2007Q4	856	20	280	1,156	76	2	74	876
3	Male	European C	2008Q1	863	25	284	1,172	76	3	74	888
4	Male	European C	2008Q2	850	26	281	1,157	76	3	74	876
5	Male	European C	2008Q3	840	30	286	1,155	75	3	73	869
6	Male	European C	2008Q4	855	30	275	1,159	76	3	74	884
7	Male	European C	2009Q1	845	35	279	1,160	76	4	73	880
8	Male	European C	2009Q2	832	35	280	1,146	76	4	73	866
9	Male	European C	2009Q3	813	42	290	1,146	75	5	71	856
10	Male	European C	2009Q4	831	40	277	1,148	76	5	72	871
11	Male	European C	2010Q1	822	36	283	1,142	75	4	72	859
12	Male	European C	2010Q2	825	40	290	1,155	75	5	71	865
13	Male	European C	2010Q3	837	31	287	1,155	75	4	72	868
14	Male	European C	2010Q4	838	40	277	1,155	76	4	73	878

Figure 2: An example of a LongForm dataframe. This is the Table in Figure 1 after automatic conversion with **TableToLongForm** and in total spans 660 rows. Now all related information can be found in the same row or column, making the data much more useful.

User manual

Loading the data

TableToLongForm's preferred argument is a "matrix" of mode "character". If a "data.frame" is supplied instead, it is coerced to a "matrix" with a warning. Empty cells should be classed as "NA" for correct operation of the algorithms. Currently **TableToLongForm** does not distinguish between missing values and empty space, both are treated as "NA" values.

As the Labour Force Status data used in Figure 1 classifies missing values as '.', we must ensure R correctly reads these, in addition to empty cells, as "NA" values.⁴

```
LabourForce = as.matrix(read.csv("StatsNZLabourForce.csv",
                                header = FALSE, na.strings = c("", ".")))
```

³I use the term LongForm loosely and in some cases **TableToLongForm** will result in WideForm output as the difference can depend on contextual information the computer cannot comprehend. However, the output will be machine readable and many tools, such as those mentioned in the opening paragraph, can be used to further reformat the data, including conversions between LongForm and WideForm.

⁴This Table, after being read in as a "matrix" as shown, is included in **TableToLongForm** as part of data(TCData), and can be accessed with TCData[["StatsNZLabourForce"]]

Calling `TableToLongForm`

If the Table can be recognised by `TableToLongForm`, a simple call to `TableToLongForm` with just a single argument is all that is needed. `TableToLongForm` has additional optional arguments used primarily for diagnostic purposes, which are covered in the diagnostics section at the end of the article.

```
LabourForce.converted = TableToLongForm(LabourForce)
```

Aside: manual conversion

For comparison the code for manual conversion of the table is provided below. We note after careful observation of the data that:

- There are 3 gender categories: 'Male', 'Female' and 'Total Both Sexes', each 80 columns in width.
- There are 10 ethnic categories, each a consistent 8 columns in width.
- The data are found in rows 5 to 26.

Armed with this knowledge, we can write the above code that, with some trial and error and cross-checking of results, will successfully convert the Table to a LongForm. This code is fairly compact and efficiency-wise beats `TableToLongForm`, taking a little over a thousandth of a second to make the conversion (compared to about a hundredth of a second for a call to `TableToLongForm`) on the author's machine. However, it took a non-trivial investment of time to code and test the results (it took the author about 30 minutes), is mostly useless for any other Table, and if any of the many strong assumptions it makes are violated (e.g. a new row of data is added), it breaks and requires fixing, which means even more time consumed. All this work and hassle to just *read in the data* in a useful format.

```
LFout = NULL
chYear = LabourForce[5:26, 1]
for(Gender in 0:2)
  for(Ethni in 0:9){
    chGender = LabourForce[2, 2 + Gender * 80]
    chEthni = LabourForce[3, 2 + Ethni * 8]
    LFout = rbind(LFout,
      cbind(chGender, chEthni, chYear,
        LabourForce[5:26, 2 + Gender * 80 + (Ethni * 8):((Ethni + 1) * 8 - 1)])
    )
  }
colnames(LFout) = c("Gender", "Ethnicity", "Time.Period", LabourForce[4, 2:9])
```

IdentResult

For a successful conversion, `TableToLongForm` must first find the Table, that is, it must *Identify* the rows and columns in which the labels and data values can be found. This task can be surprisingly difficult, requiring many fringe-case checks and exception handling. The current core identification algorithm searches for blocks (rectangular regions) of numbers in the supplied "matrix". This region is assumed to contain the data and from it `TableToLongForm` infers the locations of the corresponding labels. The result, after some extra work to handle fringe-cases and the like, is the `IdentResult`, a "list" which specifies the rows and columns in which the labels and the data can be found.

If `TableToLongForm` fails to correctly Identify the correct rows and columns, it is possible to manually specify the `IdentResult` as an argument. This is the case for the Table in [Figure 3](#), where one of the row label columns is a Year column consisting only of numbers. `TableToLongForm`'s numeric label detection algorithm is still quite primitive and fails to correctly identify column 3 as a label, but by manually specifying the `IdentResult`, `TableToLongForm` can still successfully convert the Table; the resulting "data.frame" is shown in [Figure 4](#). Even for cases such as this where the `IdentResult` must be manually specified, the work required for the conversion with `TableToLongForm` will be strictly less than for a manual conversion as we would need the same information, and more, to convert manually.

```
TableToLongForm(NEET, IdentResult = list(rows = list(label = 3:4, data = 5:46),
  cols = list(label = 2:3, data = 4:24)))
```

	1	2	3	4	5	6	7	8	9	10	11	
1	TABLE 1: (a) Number of 16–24 year olds Not in Education, Employment or Training (NEET) and (b) associated Confidence Intervals by											
2												
3				(a) Number								
4		Quarterly LFS series		England	North East	North West	Yorks & Hur	East Midlan	West Midlan	East of Eng	London	
5		Q2	2000	652,000	61,000	92,000	72,000	60,000	75,000	53,000	127,000	
6		Q3	2000	750,000	57,000	113,000	87,000	69,000	89,000	64,000	131,000	
7		Q4	2000	629,000	48,000	97,000	72,000	57,000	84,000	55,000	93,000	
8		Q1	2001	667,000	53,000	114,000	77,000	58,000	82,000	61,000	100,000	
9		Q2	2001	650,000	42,000	112,000	77,000	53,000	75,000	62,000	111,000	
10		Q3	2001	774,000	50,000	132,000	84,000	60,000	79,000	75,000	140,000	
11		Q4	2001	660,000	46,000	110,000	75,000	50,000	79,000	60,000	116,000	
12		Q1	2002	699,000	51,000	114,000	83,000	59,000	88,000	61,000	111,000	
13		Q2	2002	703,000	45,000	117,000	84,000	55,000	85,000	61,000	123,000	
14		Q3	2002	795,000	49,000	115,000	111,000	58,000	96,000	71,000	143,000	

Figure 3: Another example of a hierarchical Table. The Table is of the NEET statistics (Department for Education (UK), 2013) and is relatively tame in terms of complexity. The work required to manually convert and read in such a Table would be light, but still enough to be an annoying time sink. Highlighted are the three regions TableToLongForm must identify for successful conversion, and if automatic identification of these regions fail, the rows and columns corresponding to these three rectangular regions can be specified manually.

	1	2	3	4	5	6	7	8	9	10	11
1				England	North East	North West	Yorks & Hur	East Midlan	West Midlan	East of Eng	London
2	(a) Number	Q2	2000	652,000	61,000	92,000	72,000	60,000	75,000	53,000	127,000
3	(a) Number	Q3	2000	750,000	57,000	113,000	87,000	69,000	89,000	64,000	131,000
4	(a) Number	Q4	2000	629,000	48,000	97,000	72,000	57,000	84,000	55,000	93,000
5	(a) Number	Q1	2001	667,000	53,000	114,000	77,000	58,000	82,000	61,000	100,000
6	(a) Number	Q2	2001	650,000	42,000	112,000	77,000	53,000	75,000	62,000	111,000
7	(a) Number	Q3	2001	774,000	50,000	132,000	84,000	60,000	79,000	75,000	140,000
8	(a) Number	Q4	2001	660,000	46,000	110,000	75,000	50,000	79,000	60,000	116,000
9	(a) Number	Q1	2002	699,000	51,000	114,000	83,000	59,000	88,000	61,000	111,000
10	(a) Number	Q2	2002	703,000	45,000	117,000	84,000	55,000	85,000	61,000	123,000
11	(a) Number	Q3	2002	795,000	49,000	115,000	111,000	58,000	96,000	71,000	143,000
12	(a) Number	Q4	2002	660,000	49,000	100,000	74,000	55,000	77,000	69,000	113,000
13	(a) Number	Q1	2003	730,000	51,000	99,000	95,000	54,000	90,000	78,000	122,000
14	(a) Number	Q2	2003	709,000	51,000	107,000	87,000	59,000	88,000	66,000	112,000

Figure 4: Another example of a LongForm dataframe. This is the Table in Figure 3 after automatic conversion with TableToLongForm. Although the conversion required the aid of a human to specify the optional argument IdentResult to be successful, the work required with TableToLongForm will be strictly less than for a manual conversion as we would need the same information, and more, to convert manually.

Recognised patterns

TableToLongForm consists of a number of algorithms that can collectively process a variety of so-called *recognised patterns* of hierarchical structure (also called the *parentage* of the labels). Any Table that consists of some combination of the recognised patterns can be automatically converted with TableToLongForm. It is not strictly necessary for a user to know what the patterns are, as they can simply try calling TableToLongForm on the Table to see if it converts. All the recognised patterns are listed here for reference.⁵ For an example of a real Table that demonstrates a combination of the recognised patterns, refer to **Real Example - NZQA** located at the end of this section.

For each pattern an example table is first shown using toy data, that displays the pattern, followed by a short description of the pattern, and ending with the example table converted with TableToLongForm.

Many of the recognised patterns apply only for row labels. Column labels are recognised by noticing that the transpose of column labels can often be processed as row labels, though there are several fringe cases that must be corrected for.

⁵All the Tables demonstrating the recognised patterns are included in TableToLongForm as part of data(TCData). TableToLongForm can be called on these Tables for the converted versions, e.g. TableToLongForm(TCData[["ToyExByEmptyBelow"]])

Empty Below

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	10	20	30	40
3		Row Child2	11	21	31	41
4	Row Parent2	Row Child1	12	22	32	42
5		Row Child2	13	23	33	43

Above, we have an example of the Empty Below pattern, the most simple type of parentage. Here the *parent* and *children* are in different columns and we can see which of the children belong to which parent through the use of empty space below each parent. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	10	20	30	40
3	Row Parent1	Row Child2	11	21	31	41
4	Row Parent2	Row Child1	12	22	32	42
5	Row Parent2	Row Child2	13	23	33	43

Empty Right 1

	1	2	3	4	5	6	7
1				Column 1	Column 2	Column 3	Column 4
2	Row Parent1			10	20	30	40
3	Row Child1	Row Child-Child1		11	21	31	41
4	Row Child2	Row Child-Child2		12	22	32	42
5	Row Parent2			13	23	33	43
6	Row Child1	Row Child-Child1		14	24	34	44
7		Row Child-Child2		15	25	35	45

Above, we have an example of the most basic form of the Empty Right pattern. In this situation we have children in the same column as their parent. We can still recognise these as children if the children have children (*Child-Child*) in a different column, while the parent does not (and hence the parent is Empty Right). Note the values pertaining to the Parent (if any) are discarded. This is because they are assumed to simply represent the sum of their children’s values. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7
1				Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	Row Child-Ch	11	21	31	41
3	Row Parent1	Row Child2	Row Child-Ch	12	22	32	42
4	Row Parent2	Row Child1	Row Child-Ch	14	24	34	44
5	Row Parent2	Row Child1	Row Child-Ch	15	25	35	45

Empty Right 2

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1		10	20	30	40
3		Row Child1	11	21	31	41
4		Row Child2	12	22	32	42
5	Row Parent2		13	23	33	43
6		Row Child1	14	24	34	44
7		Row Child2	15	25	35	45

Above, we have an example of both Empty Below and Empty Right. Either algorithm can handle this situation, but simply due to the ordering of the algorithms such situations are handled as Empty Right. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6
1			Column 1	Column 2	Column 3	Column 4
2	Row Parent1	Row Child1	11	21	31	41
3	Row Parent1	Row Child2	12	22	32	42
4	Row Parent2	Row Child1	14	24	34	44
5	Row Parent2	Row Child2	15	25	35	45

Empty Right 3

	1	2	3	4	5	6	7	8
1				Column 1	Column 2	Column 3	Column 4	
2	Row Super-Parent1			10	20	30	40	
3	Row Parent1			11	21	31	41	
4	Row Child1	Row Child-Child1		12	22	32	42	
5	Row Parent2			13	23	33	43	
6	Row Child1	Row Child-Child1		14	24	34	44	
7	Row Super-Parent2			15	25	35	45	
8	Row Parent1			16	26	36	46	
9	Row Child1	Row Child-Child1		17	27	37	47	
10	Row Parent2			18	28	38	48	
11	Row Child1	Row Child-Child1		19	29	39	49	

Above, we have an example of a complex version of the Empty Right pattern. The “parent-child in the same column” situation has been extended further and we now have parents (*Super-Parent*) who have children (*Parent*), who each further have children (*Child*), all in the same column. Such situations can still be recognised if the lowest-level children in the column (*Child*) have children in a different column (*Child-Child*), while its direct parents (*Parent*) each have children in the same column (*Child*) but not in a different column (is Empty Right), and the top-most parents (*Super-Parents*) also have no children in a different column (is also Empty Right). The algorithm cannot currently handle super-super-parents. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8
1					Column 1	Column 2	Column 3	Column 4
2	Row Super-P	Row Parent1	Row Child1	Row Child-Ch	12	22	32	42
3	Row Super-P	Row Parent2	Row Child1	Row Child-Ch	14	24	34	44
4	Row Super-P	Row Parent1	Row Child1	Row Child-Ch	17	27	37	47
5	Row Super-P	Row Parent2	Row Child1	Row Child-Ch	19	29	39	49

Multi-row Column Label

	1	2	3	4	5	6
1		Column	Column	Column	Column	
2		Child1	Child2	Child3	Child4	
3	Row 1	10	20	30	40	
4	Row 2	11	21	31	41	
5	Row 3	12	22	32	42	
6	Row 4	13	23	33	43	

Above, we have an example of Multi-row Column Labels. Often column labels are physically split over multiple rows rather than making use of line breaks in the same cell. In such occurrences, any row not identified as a parent are collapsed into a single row of labels. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6
1		Column Child1	Column Child2	Column Child3	Column Child4	
2	Row 1	10	20	30	40	
3	Row 2	11	21	31	41	
4	Row 3	12	22	32	42	
5	Row 4	13	23	33	43	

Mismatched Column Label

	1	2	3	4	5	6	7	8	9
1		Col Child1		Col Child2		Col Child3		Col Child4	
2	Row 1		10		20		30		40
3	Row 2		11		21		31		41
4	Row 3		12		22		32		42
5	Row 4		13		23		33		43

Above, we have an example of Mismatched Column Labels. Sometimes the column labels are in a different column to the data, usually due to a misguided attempt at visual alignment of labels to the data. As long as the correct rows and columns were identified for the data and the labels (see [User manual](#) subsection on `IdentResult`), and if there are the same number of data columns as label columns, these mismatched column labels will be paired with the data columns. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9
1		Col Child1	Col Child2	Col Child3	Col Child4				
2	Row 1	10	20	30	40				
3	Row 2	11	21	31	41				
4	Row 3	12	22	32	42				
5	Row 4	13	23	33	43				

Misaligned Column Label

	1	2	3	4	5	6	7	8	9
1			Col Parent1				Col Parent2		
2		Col Child1	Col Child2	Col Child3	Col Child4	Col Child1	Col Child2	Col Child3	Col Child4
3	Row 1	10	20	30	40	50	60	70	80
4	Row 2	11	21	31	41	51	61	71	81
5	Row 3	12	22	32	42	52	62	72	82
6	Row 4	13	23	33	43	53	63	73	83

Above, we have an example of Misaligned Column Labels. Often column parents are physically centred over their children (N.B. where a spreadsheet’s cell-merge feature is used to do the centering, the actual value is usually stored in the top-left cell and hence causes no problems). **TableToLongForm** makes use of pattern recognition to identify repeating patterns in the labels, or in empty cells surrounding the labels, to correct for the misalignment. For the *Column Parents* row, we find (starting from column 2, the first data column) a pattern of Empty-NonEmpty-Empty-Empty, with the pattern occurring twice. In the *Col Child* row, we also find a pattern of length 4 occurring twice. This can be used to correctly align the *Column Parents* to its children. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9
1			Col Child1	Col Child2	Col Child3	Col Child4			
2	Col Parent1	Row 1	10	20	30	40			
3	Col Parent1	Row 2	11	21	31	41			
4	Col Parent1	Row 3	12	22	32	42			
5	Col Parent1	Row 4	13	23	33	43			
6	Col Parent2	Row 1	50	60	70	80			
7	Col Parent2	Row 2	51	61	71	81			
8	Col Parent2	Row 3	52	62	72	82			
9	Col Parent2	Row 4	53	63	73	83			

Misaligned Column Label 2

	1	2	3	4	5	6	7	8	9
1				Col Super-Parent					
2			Col Parent1				Col Parent2		
3		Col Child1	Col Child2	Col Child3	Col Child4	Col Child1	Col Child2	Col Child3	Col Child4
4	Row 1	10	20	30	40	50	60	70	80
5	Row 2	11	21	31	41	51	61	71	81
6	Row 3	12	22	32	42	52	62	72	82
7	Row 4	13	23	33	43	53	63	73	83

Above, we have a generalised example of Misaligned Column Labels. We now have *Column Super-Parent* which is misaligned to both its direct children, the *Column Parents*, and to the lowest-level children. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9
1				Col Child1	Col Child2	Col Child3	Col Child4		
2	Col Super-Pa	Col Parent1	Row 1	10	20	30	40		
3	Col Super-Pa	Col Parent1	Row 2	11	21	31	41		
4	Col Super-Pa	Col Parent1	Row 3	12	22	32	42		
5	Col Super-Pa	Col Parent1	Row 4	13	23	33	43		
6	Col Super-Pa	Col Parent2	Row 1	50	60	70	80		
7	Col Super-Pa	Col Parent2	Row 2	51	61	71	81		
8	Col Super-Pa	Col Parent2	Row 3	52	62	72	82		
9	Col Super-Pa	Col Parent2	Row 4	53	63	73	83		

Real Example - NZQA

	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	Scholarship Entries and Results by Gender and Ethnicity (Broken down by Decile)													
2														
3				Decile 1-3									Decile 4-7	
4			# of	#	#		#	# Not	#	#		# of	#	
5	Results		Entries	Absent	SNA		Assessed	Achieved	Scholarship	Outstanding		Entries	Absent	
6														
7	All Subjects		714	148	13		553	462	81	10		6,482	1,772	
8														
9	Accounting		22	4	0		18	16	2	0		156	41	
10	NZ Maori	Male	2	1	0		1	1	0	0		2	1	
11		Female	0	0	0		0	0	0	0		7	2	
12	NZ Europe	Male	2	0	0		2	1	1	0		51	13	
13		Female	3	0	0		3	2	1	0		44	12	
14		Unknown	0	0	0		0	0	0	0		0	0	
15	Pasifika Pe	Male	2	0	0		2	2	0	0		3	0	
16		Female	6	2	0		4	4	0	0		4	2	
17	Asian	Male	5	0	0		5	5	0	0		29	4	
18		Female	2	1	0		1	1	0	0		15	7	
19	Other/Unsp	Male	0	0	0		0	0	0	0		0	0	
20		Female	0	0	0		0	0	0	0		1	0	
21														
22	Agricultural & Horticultu		0	0	0		0	0	0	0		15	3	
23	NZ Maori	Male	0	0	0		0	0	0	0		0	0	
24		Female	0	0	0		0	0	0	0		0	0	
25	NZ Europe	Male	0	0	0		0	0	0	0		10	1	

Above, we have an example of real data released in a Table⁶ that demonstrates a combination of many of the patterns listed above. The data comes from the New Zealand Qualifications Authority (NZQA, 2012) regarding the entries and results for their Scholarship Exams. The Table demonstrates Empty Below, Empty Right (including Type 3, as 'All Subjects' is a *super-parent*), Multi-row Column Labels, and Misaligned Column Labels. The Table is substantially more complex than the Labour Force Status data used in Figure 1, and will require considerably more work to convert manually. Worse, each year of the data is stored in a separate Table, each with slight differences in format. Thus the manual conversion code would have to either be individually tweaked for each year (requiring yet more work), or be flexible enough to handle these differences (requiring substantially more work). Other data from NZQA faces bigger problems; though the Scholarships data can all be obtained in a mere 8 Tables (for 8 years from 2004 to 2011), the Subjects data not only requires a separate Table for each year, but also for each subject (of which there are 91). Obtaining the greatest breakdown possible for the Subjects data across all other variables requires thousands of individual Tables. Without automatic conversion with **TableToLongForm**, simply reading in such data for use would require too much work to be practical. The Table after conversion to a LongForm follows.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1						# of Entries	# Absent	# SNA	# Assessed	# Not Ach	# Scholars	# Outstanding	
2	Decile 1-3	All Subjects	Accounting	NZ Maori	Male	2	1	0	1	1	0	0	
3	Decile 1-3	All Subjects	Accounting	NZ Maori	Female	0	0	0	0	0	0	0	
4	Decile 1-3	All Subjects	Accounting	NZ Europe	Male	2	0	0	2	1	1	0	
5	Decile 1-3	All Subjects	Accounting	NZ Europe	Female	3	0	0	3	2	1	0	
6	Decile 1-3	All Subjects	Accounting	NZ Europe	Unknown	0	0	0	0	0	0	0	
7	Decile 1-3	All Subjects	Accounting	Pasifika Pe	Male	2	0	0	2	2	0	0	
8	Decile 1-3	All Subjects	Accounting	Pasifika Pe	Female	6	2	0	4	4	0	0	
9	Decile 1-3	All Subjects	Accounting	Asian	Male	5	0	0	5	5	0	0	
10	Decile 1-3	All Subjects	Accounting	Asian	Female	2	1	0	1	1	0	0	
11	Decile 1-3	All Subjects	Accounting	Other/Unsp	Male	0	0	0	0	0	0	0	
12	Decile 1-3	All Subjects	Accounting	Other/Unsp	Female	0	0	0	0	0	0	0	
13	Decile 1-3	All Subjects	Agricultural	NZ Maori	Male	0	0	0	0	0	0	0	
14	Decile 1-3	All Subjects	Agricultural	NZ Maori	Female	0	0	0	0	0	0	0	
15	Decile 1-3	All Subjects	Agricultural	NZ Europe	Male	0	0	0	0	0	0	0	
16	Decile 1-3	All Subjects	Agricultural	NZ Europe	Female	0	0	0	0	0	0	0	
17	Decile 1-3	All Subjects	Agricultural	NZ Europe	Unknown	0	0	0	0	0	0	0	
18	Decile 1-3	All Subjects	Agricultural	Pasifika Pe	Male	0	0	0	0	0	0	0	
19	Decile 1-3	All Subjects	Agricultural	Pasifika Pe	Female	0	0	0	0	0	0	0	
20	Decile 1-3	All Subjects	Agricultural	Asian	Male	0	0	0	0	0	0	0	
21	Decile 1-3	All Subjects	Agricultural	Asian	Female	0	0	0	0	0	0	0	
22	Decile 1-3	All Subjects	Agricultural	Other/Unsp	Male	0	0	0	0	0	0	0	
23	Decile 1-3	All Subjects	Agricultural	Other/Unsp	Female	0	0	0	0	0	0	0	
24	Decile 1-3	All Subjects	Art History	NZ Maori	Male	0	0	0	0	0	0	0	
25	Decile 1-3	All Subjects	Art History	NZ Maori	Female	2	0	0	2	2	0	0	

⁶This Table is included in **TableToLongForm** as part of data(TCData), and can be accessed with TCData[["NZQAScholarships"]]

Diagnostics

The primary limitation of **TableToLongForm** is that the function will be a black box to most users. After running the function on a Table, the user will either be given back a "data.frame" with no easy way of verifying if the result is correct, or be confronted with an error with little idea of what went wrong. Based on ad hoc tests conducted so far, **TableToLongForm** will either succeed, or fail catastrophically in a manner that is easily recognised as utter failure. However methods for verifying correct operation (or to understand failures) would be desirable.

The simplest method currently available is to examine the additional output returned when **TableToLongForm** is called with the optional argument `fulloutput = TRUE`. This will return the 'final product' of **TableToLongForm**'s algorithms in the form of 'IdentResult', 'rowplist' and 'colplist'.

IdentResult was covered above in the [User manual](#) section and contains information on where the data and labels are found.

rowplist and **colplist** stand for Row/Column Parentage List which are nested "list" objects that represents all the hierarchical relationships in the Table, as identified by **TableToLongForm**. For easier reading they are assigned the "plist" class which has a custom 'print' method. An example of a 'colplist' is shown in [Figure 5](#).

```
> TableToLongForm(LabourForce, fulloutput = TRUE)[["colplist"]]
+ Male (1, 2)
- + European Only (1, 3)
- - + Persons Employed in Labour Force (1, 4)
- - + Persons Unemployed in Labour Force (2, 4)
- - + Not in Labour Force (3, 4)
- - + Working Age Population (4, 4)
- - + Labour Force Participation Rate (5, 4)
- - + Unemployment Rate (6, 4)
- - + Employment Rate (7, 4)
- - + Total Labour Force (8, 4)
- + Maori Only (9, 3)
- - + Persons Employed in Labour Force (9, 4)
## Output truncated
```

Figure 5: A truncated example of the 'colplist' for the Labour Force Status data used in [Figure 1](#). It represents the hierarchical relationships of the column labels, as identified by **TableToLongForm**. We can see that it has correctly identified 'Male' as a top-level parent with the ethnic categories, such as 'European Only', nested inside, which are in turn a parent to the lowest-level categories, such as 'Employment Rate'.

Unfortunately this output has two key limitations. First, it is not obvious from this output what went wrong (or if nothing went wrong), requiring some detective work to piece together the evidence. Second, if anything did go wrong, the user still does not know *why*.

The option with the potential to provide the most information is calling **TableToLongForm** with the optional argument `diagnostics = TRUE`, which will write diagnostic output to a file, printing key variables at each major stage of the conversion process. This output can thus be used to track **TableToLongForm**'s progress as it works to convert the Table, enabling fairly accurate assessment of where exactly it went wrong. Some example diagnostic output is shown in [Figure 6](#). Unfortunately, understanding this output requires familiarity with the workings of the code and is unlikely to be of much use to anyone other than the author.

Discussion

This article has introduced **TableToLongForm**, an R package that can automatically convert hierarchical Tables that would normally rely on the discerning powers of a human brain, to a simple LongForm dataframe that any decent software package can easily manipulate and use. While it can handle a variety of Tables automatically, and an even greater variety with some aid from the human user, it is not without limitations. Ultimately, **TableToLongForm** still uses algorithms to detect a known set of recognised patterns and any Table that deviates from these patterns will break **TableToLongForm**.

There is work to be done in refining and generalising existing algorithms and creating new algorithms so that **TableToLongForm** can successfully handle more cases, while also reducing the


```

###TCR CIMCB rowData
[1] 5 26
###TCR CIMCB colData
[1] 2 241
###TCR IOOC plist
$rows
[1] 1 2 3 4 5 6 7 8
$cols
[1] 4
###TCR IOOC res
+ Persons Employed in Labour Force (1, 4)
+ Persons Unemployed in Labour Force (2, 4)
+ Not in Labour Force (3, 4)
+ Working Age Population (4, 4)
+ Labour Force Participation Rate (5, 4)
+ Unemployment Rate (6, 4)
+ Employment Rate (7, 4)
+ Total Labour Force (8, 4)

```

Figure 6: A few examples of the diagnostic output generated by `TableToLongForm` when called with `'diagnostics = TRUE'` on the Labour Force Status data used in [Figure 1](#). The diagnostic output is printing key variables at each major stage of the conversion process. `'###TCR'` indicates an identifier line, the following word indicates the part of the function generating this output (e.g. `'CIMCB'`, which is short for Call Identification algorithm Most-Common-Boundary), the last word indicates the name of the variable being printed (e.g. `'rowData'`). The diagnostic output can be used to see what `TableToLongForm` is getting right (or wrong)... assuming the user is familiar with the code.

possibility of a false positive. These range from adding more robust checks to the algorithms to verify correct detection, such as sum-checks or pattern recognition, to more fundamental changes, such as altering the "NA" classification to distinguish between empty space and missing values. A recent addition to the package (introduced in version 1.3.0) is to enable new, custom algorithms to be used in place of the default ones included with the package. More information on this can be found in the official webpage for the package.

There is also work to be done in diagnostics output, not only in the formal diagnostic output, but also in the form of error and warning messages. Consider for instance the following error message if we call `TableToLongForm` on the Table in [Figure 3](#) without specifying the correct `IdentResult`. From these messages it is not at all obvious that the problem is an incorrect `IdentResult`, which is a problem that is relatively easy to address if only it can correctly be identified by the user.

```

Error in 1:ncol(datbit) : argument of length 0
In addition: Warning message:
In rbind(matColLabel[!fullrows, , drop = FALSE], collapsedlabels) :
  number of columns of result is not a multiple of vector length (arg 2)

```

In terms of formal diagnostic output, various ideas are being tried such as graphical representations of the information provided by `fulloutput = TRUE` by drawing the original Table with the regions being highlighted in some way. Such a method would, for example, make it easier to see a problem with `IdentResult`, as it should become apparent on the drawn Table that the incorrect regions are being highlighted.

This article has been written for `TableToLongForm` version 1.3.1. The code for reproducing the figures in this article, as well as more detailed documentation on the code itself, can be found at <https://www.stat.auckland.ac.nz/~joh024/Research/TableToLongForm/>. The development version can be found on github at <https://github.com/joh024/TableToLongForm>.

Bibliography

- Department for Education (UK). NEET statistics quarterly brief: April to June, 2013. URL <https://www.gov.uk/government/publications/neet-statistics-quarterly-brief-april-to-june-2013>. [p19]
- NZQA. Secondary School Statistics, 2012. URL <http://www.nzqa.govt.nz/>. [p23]
- OpenRefine. OpenRefine, 2013. URL <http://openrefine.org/>. [p16]
- Statistics New Zealand. Infoshare, 2013. URL <http://www.stats.govt.nz/infoshare/>. [p17]
- I. Visne, A. Yildiz, E. Dilaveroglu, K. Vierlinger, C. Nöhammer, F. Leisch, and A. Kriegner. speedR: An R package for interactive data import, filtering and ready-to-use code generation. *Journal of Statistical Software*, 51(2):1–12, 2012. URL <http://www.jstatsoft.org/v51/i02/>. [p16]
- H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL <http://www.jstatsoft.org/v21/i12/>. [p16]

- H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1–29, 2011. URL <http://www.jstatsoft.org/v40/i01/>. [p16]
- H. Wickham. Tidy data. *Journal of Statistical Software*, 59(10), 9 2014. ISSN 1548-7660. URL <http://www.jstatsoft.org/v59/i10>. [p16]

Jimmy Oh
Department of Statistics
The University of Auckland
New Zealand
joh024@aucklanduni.ac.nz

Prinsimp

by Jonathan Zhang, Nancy Heckman, Davor Cubranic, Joel G. Kingsolver, Travis Gaydos and J.S. Marron

Abstract Principal Components Analysis (PCA) is a common way to study the sources of variation in a high-dimensional data set. Typically, the leading principal components are used to understand the variation in the data or to reduce the dimension of the data for subsequent analysis. The remaining principal components are ignored since they explain little of the variation in the data. However, the space spanned by the low variation principal components may contain interesting structure, structure that PCA cannot find. **Prinsimp** is an R package that looks for interesting structure of low variability. “Interesting” is defined in terms of a simplicity measure. Looking for interpretable structure in a low variability space has particular importance in evolutionary biology, where such structure can signify the existence of a genetic constraint.

Principal component analysis

Principal Components Analysis is simply an eigenanalysis of a covariance or correlation matrix. To learn more about PCA, see any standard multivariate analysis text, such as [Johnson and Wichern \(2007\)](#). Through an eigenanalysis, a $d \times d$ covariance matrix G can be decomposed in terms of its eigenvectors, v_1, \dots, v_d , and the corresponding eigenvalues, $\lambda_1, \dots, \lambda_d$: $G = \sum_1^d \lambda_j v_j v_j^T$. The eigenvalue, λ_j , can be interpreted as the variance in G in the direction of v_j and $\lambda_j / \sum_{k=1}^d \lambda_k$ as the proportion of variance explained by v_j . Often, G is well-approximated using the first J eigenvectors and eigenvalues. Typically, data analysts only consider these J eigenvectors, and for that reason, we call the space that these eigenvectors span the *model space*. We call the orthogonal complement of the model space (which is simply the space spanned by the last $d - J$ eigenvectors) the *nearly null space*.

Researchers can use existing R functions `princomp` and `prcomp` to calculate the eigenvectors, eigenvalues and proportion of variance explained. Likewise, our library `prinsimp` ([Cubranic et al., 2013](#)) can be used to study eigenvectors and their properties. In addition, `prinsimp` allows researchers to more carefully study the nearly null space, by calculating simple and easily interpretable basis vectors.

We focus on analysis of covariance matrices, although the user can input a correlation matrix for analysis. As in usual PCA, the choice between using a correlation matrix and a covariance matrix depends upon the types of responses and the question of interest. A correlation matrix should be used when the responses under consideration have non-comparable units of measurement, such as meters for measuring the response, height, and kilograms for measuring the response, weight. In all of our applications, the responses are directly comparable: they are from the same variable measured under different conditions (e.g., the variable weight measured at different ages). To study structure in such data, analysis of the covariance matrix is most useful. In addition, such data usually have natural simplicity measures, whereas non-comparable data do not.

Details of the methodology that `prinsimp` implements appear in [Gaydos et al. \(2013\)](#), a paper which also discusses the importance of studying genetic constraints.

Simplicity and the simplicity basis

The simplicity of a vector $v \in \mathbb{R}^d$ is defined in terms of a non-negative definite symmetric $d \times d$ matrix Λ : the simplicity of v is equal to $v^T \Lambda v$, with large values meaning v is simple. This simplicity measure allows us to construct the *simplicity basis* of the $L = d - J$ dimensional nearly null space.

A simplicity basis for an L -dimensional linear subspace \mathcal{V} of \mathbb{R}^d is an orthonormal basis $\{w_1, \dots, w_L\}$ where the w_k 's are in decreasing order of simplicity, as defined by Λ . To calculate a simplicity basis, first let u_1, \dots, u_L be an orthonormal basis of \mathcal{V} and let P be the $d \times L$ matrix with k th column equal to u_k . So P is a projection matrix onto \mathcal{V} . Let a_1, \dots, a_L be the eigenvectors of $P^T \Lambda P$ with corresponding eigenvalues $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_L$. Then it is straightforward to show that $\{Pa_1, \dots, Pa_L\}$ is a simplicity basis of \mathcal{V} . If the eigenvalues of Λ are distinct, then any orthonormal basis $\{u_1, \dots, u_L\}$ will lead to the same set of $\{w_1, \dots, w_L\}$. For our purposes, we set $\{u_1, \dots, u_L\}$ equal to the last L eigenvectors of G .

Note that we require that simple v 's have large values of $v^T \Lambda v$. However, in some cases, the simplicity measure is most easily defined in terms of a non-negative definite matrix Λ^* with simple v 's having *small* values of $v^T \Lambda^* v$. In such a case, we simply set $\Lambda = \lambda^* I - \Lambda^*$ where λ^* is the largest eigenvalue of Λ^* . Then Λ is non-negative definite and simple v 's will have large values of $v^T \Lambda v$, as

desired.

Examples of quadratic simplicity measures can be found in smoothing and penalized regression. See [Eilers and Marx \(1996\)](#) and [Green and Silverman \(1994\)](#). Different examples of simplicity measures are discussed in the next section, in particular the built-in measures of the `simpart` function.

Examples of simplicity measures

The function `simpart` has three built-in simplicity measures of vectors in \mathbb{R}^d : the default measure based on first divided differences as defined in [Gaydos et al. \(2013\)](#), another measure based on second divided differences and a third based on periodicity. These three measures are defined in terms of a vector $x \in \mathbb{R}^d$ containing values of an explanatory variable. The periodicity simplicity measure is most natural when x contains times and the researcher believes that responses are periodic or approximately periodic with known period.

Simplicity measure of a vector v using divided differences based on x

Divided differences are often used to approximate derivatives of a function. For instance, the first divided difference of the differentiable function f using x_1 and x_2 is $[f(x_2) - f(x_1)]/[x_2 - x_1]$, which serves as an approximation of $f'(x_1)$. Our divided difference simplicity measures are appropriate when the observed data can be considered as arising from function evaluations.

To define a divided difference simplicity measure of a vector based on the vector of explanatory variables, $x = (x[1], \dots, x[d])^T$, let the $(d-1) \times d$ difference matrix D_d be

$$D_d = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

and let W be the $(d-1) \times (d-1)$ diagonal matrix with $W[j, j] = x[j+1] - x[j]$, that is, with the diagonal of W equal to $D_d x$.

First divided difference simplicity measure of v based on x

The first divided difference simplicity measure of the vector $v = (v[1], \dots, v[d])^T$ is the default measure in `simpart`. To define it, write

$$\sum_{j=1}^{d-1} \frac{(v[j+1] - v[j])^2}{x[j+1] - x[j]} = \|W^{-1/2} D_d v\|^2 = v^T D_d^T W^{-1} D_d v.$$

This expression is a *complexity measure*, with large values indicating that v is complex. However, we will define a *simplicity measure*, with large values indicating that v is simple. In addition, we will scale our measure using a result of [Schatzman \(2002\)](#), that the eigenvalues of $D_d^T W^{-1} D_d$ are always between 0 and $4/\Delta_x$, where $\Delta_x \equiv \min_j x[j+1] - x[j]$. We define the simplicity of v as $v^T \Lambda_1 v$ where $\Lambda_1 = 4I - \Delta_x D_d^T W^{-1} D_d$. Then the value of this simplicity measure is always between 0 and 4 and the simplest vector v satisfies $\sum_{j=1}^{d-1} (v[j+1] - v[j])^2 / (x[j+1] - x[j]) = 0$, that is, the components of v are equal.

Second divided difference simplicity measure of v based on x

To define the simplicity measure based on second divided differences, let

$$\delta_j = \frac{v[j+2] - v[j+1]}{x[j+2] - x[j+1]} - \frac{v[j+1] - v[j]}{x[j+1] - x[j]}, \quad j = 1, \dots, d-2.$$

Our simplicity measure of the vector v is defined in terms of

$$\sum_{j=1}^{d-2} \frac{\delta_j^2}{x[j+2] - x[j]}. \quad (1)$$

We consider v simple if this expression is small. One can easily show that this sum of squares is equal to 0 if and only if $v[j] = \alpha + \beta x[j]$ for some α and β .

To write (1) in matrix form, write

$$\begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{d-2} \end{pmatrix} = D_{d-1}W^{-1}D_d v$$

and let W_2 be the $(d-2) \times (d-2)$ diagonal matrix with $W_2[j, j] = x[j+2] - x[j]$. Thus (1) is equal to $v^T D_d^T W^{-1} D_{d-1}^T W_2^{-1} D_{d-1} W^{-1} D_d v \equiv v^T \Lambda_2^* v$. To make this a simplicity measure, we set $\Lambda_2 = \lambda^* I - \Lambda_2^*$, with λ^* equal to the largest eigenvalue of Λ_2^* . The simplicity measure of v is $v^T \Lambda_2 v$. The function `simpart` calculates λ^* numerically, via eigenanalysis of Λ_2^* .

Periodicity simplicity measure of v using x

In some applications, responses may vary periodically. For instance, the growth of a plant may have a daily cycle caused by the natural daily cycle of sunlight. However, this periodic variation may be swamped by larger sources of variation and thus not detectable by Principal Components Analysis. In this case, we need new methods to look for small-variation periodic structure, that is, to look for periodic structure in the nearly null space.

To use `prinsimp`'s periodic simplicity measure, the user must input a period. For example, if plant height had a daily cycle and was measured hourly, then the input period would be 24. Our periodic simplicity measure is equal to 1 if and only if the plant heights are exactly periodic. The definition of the simplicity measure is somewhat technical, and so we provide it in the following separate section, for the interested reader.

Details of the definition of the periodic simplicity measure of v using x

As noted, to define a periodicity measure of a vector, the user must specify an *index period* p , a positive integer. We say that $v = (v[1], \dots, v[d])^T$ is periodic with index period p if and only if $v[j+p] = v[j]$, $j = 1, d-p$.

This definition of periodicity only makes practical sense if v is a vector of responses corresponding to a vector $x = (x[1], \dots, x[d])^T$ with the $x[j]$'s being time points that "match" the index period. That is, to make practical sense, the data analyst would want x to support periodicity with time period π and index period p : that $x[j+p] = x[j] + \pi$, $j = 1, \dots, d-p$. However, the formal definitions that follow do not require the vector x .

The periodic simplicity measure of a vector $v \in \mathbb{R}^d$ for a given index period p is defined as follows. Write $d = lp + k$ with $0 \leq k < p$. First suppose that $k = 0$, so that d is divisible by p . Then we can partition the vector v into l segments each of length p : $v^T \equiv (v^{(1)T}, \dots, v^{(l)T})$. Let $\bar{v}^* \in \mathbb{R}^p$ be the component-wise average of the $v^{(i)}$'s. So \bar{v}^* 's j th component is

$$\bar{v}^*[j] = \frac{1}{l} \sum_{i=1}^l v^{(i)}[j].$$

The periodic simplicity measure of v , a vector of length $d = lp$, is defined in terms of the complexity measure

$$\sum_{i=1}^l \|v^{(i)} - \bar{v}^*\|^2. \tag{2}$$

This measure is zero if and only if the vector v is periodic with index period p , that is, if and only if $v[j+p] = v[j]$ for all $j = 1, \dots, d-p$.

If the length of v is $d = lp + k$ with $0 < k < p$, then we divide v into $l+1$ segments $v^T \equiv (v^{(1)T}, \dots, v^{(l)T}, v^{(l+1)T})$ with $v^{(i)} \in \mathbb{R}^p$, $i = 1, \dots, l$ and $v^{(l+1)} \in \mathbb{R}^k$. We define $\bar{v}^* \in \mathbb{R}^p$ as a component-wise average, in the obvious way:

$$\bar{v}^*[j] = \begin{cases} \frac{1}{l+1} \sum_{i=1}^{l+1} v^{(i)}[j] & \text{for } 1 \leq j \leq k \\ \frac{1}{l} \sum_{i=1}^l v^{(i)}[j] & \text{for } k+1 \leq j \leq p. \end{cases}$$

The periodic simplicity measure of v , a vector of length $d = lp + k$, $0 < k < p$, is defined in terms of

the complexity measure

$$\sum_{i=1}^l \|v^{(i)} - \bar{v}^*\|^2 + \sum_{j=1}^k \left(v^{(l+1)}[j] - \bar{v}^*[j] \right)^2. \quad (3)$$

This measure is 0 if and only if $v^{(1)} = \dots = v^{(l)}$ and the components of $v^{(l+1)}$ are equal to the first k components of $v^{(1)}$, that is, if and only if v is periodic with index period p .

We can easily find Λ_π^* so that $v' \Lambda_\pi^* v$ equals the complexity measures in (2) and (3). Since these expressions are small when v is close to periodic, we let $\Lambda_\pi = \lambda^* \mathbf{I} - \Lambda_\pi^*$ where λ^* is the largest eigenvalue of Λ_π^* and set our simplicity measure of v equal to $v^T \Lambda_\pi v$. To find the largest eigenvalue of Λ_π^* , we can show that Λ_π^* is a projection matrix, that is, that $\Lambda_\pi^{*T} \Lambda_\pi^* = \Lambda_\pi^*$. So the eigenvalues of Λ_π^* are all equal to 0 or 1. Thus, $\lambda^* = 1$ and $\Lambda_\pi = \mathbf{I} - \Lambda_\pi^*$. Since $v' \Lambda_\pi^* v = 0$ if and only if v is periodic with index period p and this subspace of v 's is of dimension p , Λ_π has eigenvalues equal to 1, of multiplicity p , and eigenvalues equal to 0, of multiplicity $d - p$. Eigenvectors of Λ_π corresponding to eigenvalues equal to 1 are periodic with index period p .

For an alternative method for finding periodic structure in functional data, see [Zhao et al. \(2004\)](#).

Prinsimp specifics

The main function, `simpart`, of the package **prinsimp** is designed for analysis of a $d \times d$ covariance matrix – either supplied by the user or calculated by `simpart` from data. If the user has data consisting of responses vectors $y_j \in \mathbb{R}^d$, $j = 1, \dots, n$, the user can input the $n \times d$ matrix, y , with j th row equal to y_j^T and `simpart` will calculate the required sample covariance matrix. For some data, for instance, if the y_j 's are different lengths, the user must supply a $d \times d$ covariance matrix estimated by an external method, such as random regression analysis ([Demidenko, 2004](#)) or PACE ([Yao et al., 2005](#)). Sometimes, the user must supply a vector of independent variable values, $x \in \mathbb{R}^d$, to `simpart`.

The `simpart` function uses the covariance matrix to partition \mathbb{R}^d into two subspaces, the *model space* and the *nearly null space*. The model space is spanned by the eigenvectors of the covariance matrix with the largest eigenvalues, that is, by the leading principal components from principal component analysis (PCA). The nearly null space is the orthogonal complement of the model space and explains a small amount of the variability in the data. The user specifies the dimension of the nearly null space and thus the dimension of the model space. The **prinsimp** package and the `simpart` function help the user to look for “simple” and “interesting” structure in the nearly null space. The main output of `simpart` is two sets of orthogonal basis vectors: those that span the model space and those that span the nearly null space. The output also includes information about these basis vectors, such as their simplicity measures and the percents of variance they explain. The `plot` and `basisplot` functions allow the user to visually study `simpart`'s output.

This section guides the reader through the capabilities of **prinsimp** using analyses of two data sets. We begin with an analysis of caterpillar data, illustrating the basic `simpart` function and its default `print`, `summary` and `plot` methods and how to use the argument `reverse`. This example follows the analysis from [Gaydos et al. \(2013\)](#). The second data set is simulated using sine functions and, in the analysis, we study periodicity in the nearly null space. We use the built-in measure `periodic` and we also define a function that supplies `simpart` a tailor-made definition of periodic simplicity. At the end of this section, we give an additional example to show how to use a tailored simplicity measure. The option for the user to provide a tailored simplicity measure gives `simpart` the flexibility needed to answer a specific scientific question.

Example: caterpillar data

We start with a simple example that replicates some of the analysis from [Gaydos et al. \(2013\)](#). We use the estimated genetic covariance matrix of the caterpillar growth rate data described in [Kingsolver et al. \(2004\)](#). Growth rates (mg/hour) were recorded on 529 individuals from 35 families at six temperatures: 11, 17, 23, 29, 35, 40 degrees Centigrade. Thus, the `x` vector input to `simpart` is

```
> x.cat <- c(11, 17, 23, 29, 35, 40)
```

We have a choice of the dimension of model space $J = 6, 5, \dots, 1, 0$ with corresponding dimension of the nearly null space of `simplifiedim = 0, 1, 2, \dots, 5, 6`. We load the **prinsimp** library and access the genetic covariance matrix, `caterpillar`.

```
> library(prinsimp)
> data(caterpillar)
```

Throughout this example, we will use the default simplicity measure of first divided differences.

First consider a model space of dimension 6, so the nearly null space has dimension equal to 0.

```
> cat.sim.6 <- simpart(caterpillar, simpledim = 0, x = x.cat, cov = TRUE)
```

Since the input `caterpillar` is a covariance matrix and not the original data, we must set `cov = TRUE`. The argument `measure = "first"` is not needed because "first" is the default. When we choose `simpledim = 0`, the resulting model space basis vectors are simply the usual principal components of the covariance matrix, that is, they are the eigenvectors of `caterpillar`. Recall that, for all values of `simpledim`, the model space basis vectors are always the eigenvectors of `caterpillar` corresponding to the largest eigenvalues. As in usual principal component analysis, the components of an eigenvector are called loadings. We extend this terminology to any basis vector.

Due to rounding, the covariance matrix, `caterpillar`, has one negative eigenvalue. The function `simpart` automatically sets negative eigenvalues equal to 0, reconstructs the covariance matrix and prints a warning.

We plot our results by the command

```
> plot(cat.sim.6)
```

The resulting plot appears in Figure 1.

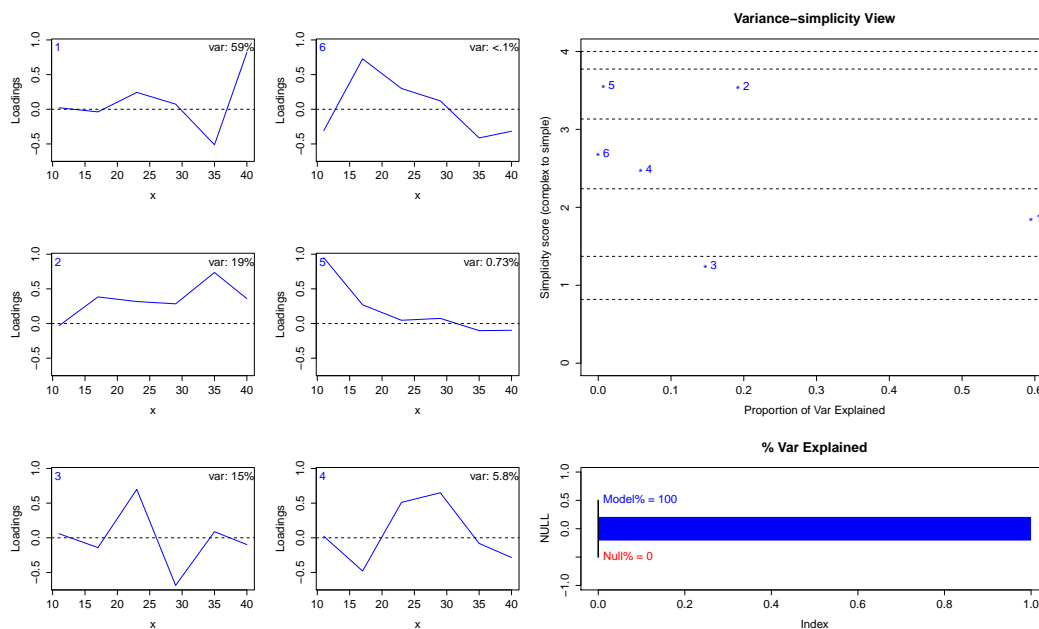


Figure 1: Analysis of the genetic covariance matrix of the caterpillar data using a model space of dimension 6 and first divided difference simplicity measure. Shown is the default output of the plot method: the six principal components of the covariance matrix and the variance-simplicity view and percent of variance explained panel.

The summary method and `display=list()` argument for the caterpillar data

Now consider the case with model space of dimension $J = 5$ and `simpledim = 1`, the dimension of the nearly null space. In this case, the nearly null space is the same space as that spanned by the sixth principal component. Thus the `simpart` output and plot will be identical to Figure 1, save for colour labelling.

```
> cat.sim.5 <- simpart(caterpillar, simpledim = 1, x = x.cat, cov = TRUE)
```

The summary method prints the simplicity measures, the percents of variance explained and the cumulative percents of variance explained of the basis vectors of both the model space and the nearly null space. If we set `loadings=TRUE`, then the summary method also prints the basis vectors of both the model space and the nearly null space.


```
> summary(cat.sim.5, loadings = TRUE)
Simple partition (first divided differences): 1 simple basis
```

	model 1	model 2	model 3	model 4	model 5	simple 1
Simplicity	1.85	3.54	1.25	2.47	3.55	2.68
%-var explained	59.5	19.2	14.7	5.85	0.731	<.1
Cumulative %-var	59.5	78.7	93.4	99.3	100	<.1

Loadings:

	model 1	model 2	model 3	model 4	model 5	simple 1
11	0.022	-0.031	0.059	0.022	0.948	-0.308
17	-0.038	0.384	-0.141	-0.479	0.270	0.727
23	0.243	0.317	0.698	0.510	0.047	0.300
29	0.074	0.284	-0.687	0.650	0.074	0.121
35	-0.512	0.736	0.088	-0.082	-0.102	-0.413
40	0.819	0.359	-0.098	-0.284	-0.098	-0.317

In calculating the percent of total variance explained, `summary` “restarts” the accumulation for the simplicity basis. In the summary output, we see, for example, that the basis vector in the 1-dimensional nearly null space has a simplicity score of 2.68 and explains less than 1% of the total variance. The cumulative percent of total variance explained is also less than 1% – this provides no additional information in this case, when the nearly null space is 1-dimensional. We know, due to the required reconstruction of the matrix caterpillar, that the percent is actually equal to 0. We can check this with the following command.

```
> cat.sim.5$variance
$model
[1] 0.618482630 0.199899686 0.153080939 0.060800542 0.007597639
$simple
[1] 2.305732e-17
$full
[1] 0.618482630 0.199899686 0.153080939 0.060800542 0.007597639 0.000000000
```

We see that `cat.sim.5$variance$simple` is essentially equal to 0 and that `cat.sim.5$variance$full[6]`, the variance associated with the sixth eigenvalue from the full eigenanalysis, is equal to 0.

We now consider a more interesting higher-dimensional nearly null space, setting `simplifiedim = 2`. The resulting model basis vectors are identical to the first four model basis vectors in Figures 1, as these are the first four principal components of caterpillar. We can make a plot similar to Figure 1, but instead we plot just the two basis vectors of the nearly null space. We do this with the `display = list()` argument in the `plot` method.

```
> cat.sim.4 <- simpart(caterpillar, simplifiedim = 2, x = x.cat, cov = TRUE)
> plot(cat.sim.4, display = list(simple = c(1, 2)))
```

This is shown in Figure 2.

Recall that the default plot, without the `display = list()` argument, would contain all of the model and nearly null space basis vectors. If the data vectors or the covariance matrix have high dimension, an unappealing number of subplots would be generated. Because of this, the `plot` method plots a maximum of six basis vectors. With the caterpillar data, this is not an issue because the data are only 6-dimensional. If the data are higher-dimensional, the user must either specify six or fewer basis vectors to plot or resort to another plotting method, such as using `basisplot`, illustrated in the next section.

The print command for the caterpillar data

The print command gives simplicity measure information: the measures of the basis vectors constructed according to the `simpart` call and the measures of the “full space” simplicity basis vectors – that is, the basis vectors of \mathbb{R}^d that would be constructed if we were to set `simplifiedim` equal to d .

```
> print(cat.sim.4)
Call:
simpart(y = caterpillar, simplifiedim = 2, x = x.cat, cov = TRUE)
```

Simplicity measure: first divided differences

Partition simplicity (2 simple basis):


```

model 1 model 2 model 3 model 4 simple 1 simple 2
1.848069 3.539756 1.245756 2.471847 3.726200 2.501706

```

Full space simplicity:

```

full 1 full 2 full 3 full 4 full 5 full 6
4.000000 3.773655 3.132870 2.237603 1.370816 0.818390

```

Note that the maximum simplicity score is 4, as claimed. We now study the percent of variance explained in the nearly null space.

```

> cat.sim.4$varperc$simple
[1] 0.6244724 0.1061671
> summary(cat.sim.4)
Simple partition (first divided differences): 2 simple basis

```

	model 1	model 2	model 3	model 4	simple 1	simple 2
Simplicity	1.85	3.54	1.25	2.47	3.73	2.50
%-var explained	59.5	19.2	14.7	5.85	0.624	0.106
Cumulative %-var	59.5	78.7	93.4	99.3	0.624	0.731

The simplest vector in the nearly null space explains 0.624% of the variance, and the two dimensional nearly null space explains 0.731% of the variance. The simplest vector is a contrast between lower and higher temperatures (see Figure 2). Interpretation of this simplest vector with respect to genetic constraints can be found in [Gaydos et al. \(2013\)](#). Other analyses of this data set, using `simplifiedim=3, 2, 1, 0`, are left to the user.

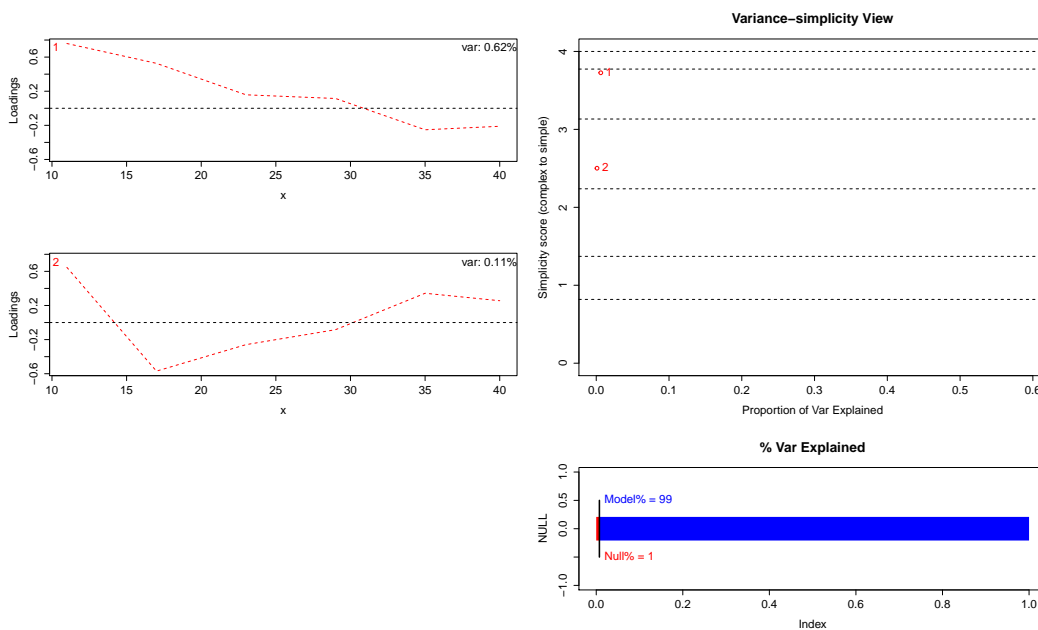


Figure 2: Analysis of the genetic covariance matrix of the caterpillar data using a model space of dimension 4 and first divided difference simplicity measure. Shown are the two basis vectors spanning the nearly null space, the variance-simplicity view and percent of variance explained panel.

The reverse argument for the caterpillar data

In principal component analysis, the signs of the loadings are arbitrary and so, in `simp`, each basis vector can have one of two possible directions. To improve interpretability the user may want to reverse the direction of a basis vector by multiplying all of its components by -1 . This can be done with the `rev` argument in `simp`. The `rev` argument takes a logical vector specifying which basis vector directions we want to reverse. For example if we would like to reverse model basis vector 2 and keep the other basis vectors' directions unchanged, we would use the following command.

```

> plot(simp(caterpillar, simplifiedim = 2, x = x.cat, cov = TRUE,
+       measure = "first",

```

```
+ rev = c(FALSE, TRUE, FALSE, FALSE, FALSE, FALSE))
> # Reverses the second basis vector, in this case the second principal component
```

This is shown in Figure 3. The subplot in the second row of the first column contains the reversed second principal component. Compare this subplot with the one in the same position in Figure 1: it is the same except for the sign of the loadings. We would not, in general, prefer the subplot in Figure 3 because all of the loadings are negative. Indeed, if we were to see such a plot in our analysis we would use the rev command to reverse its direction.

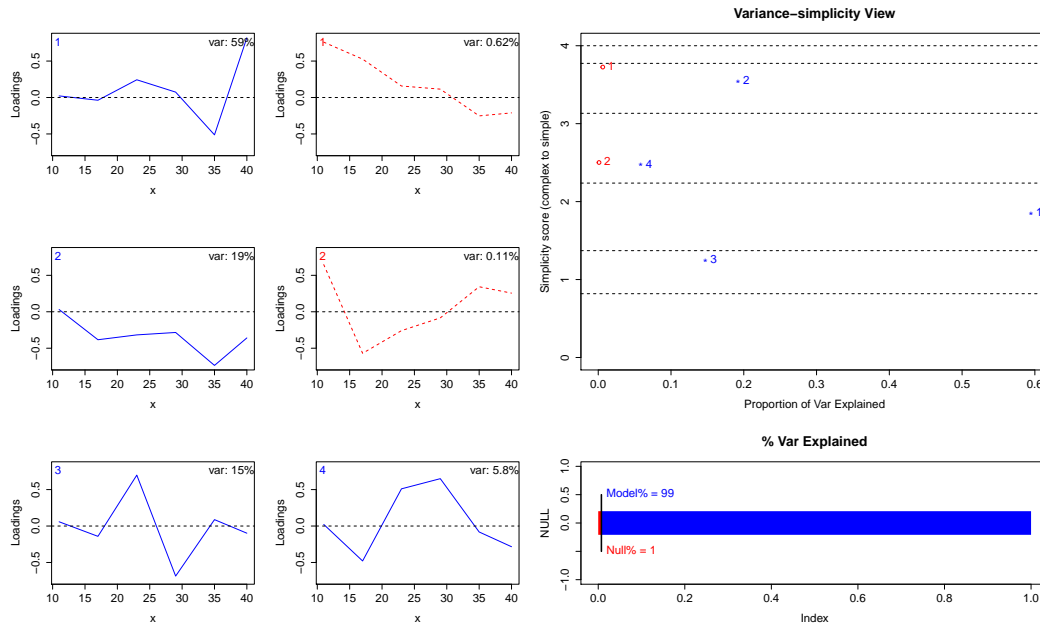


Figure 3: Analysis of the genetic covariance matrix of the caterpillar data using a model space of dimension 4 and first divided difference simplicity measure. Shown are the four basis vectors spanning the model space, the two simplicity basis vectors spanning the nearly null space, the variance-simplicity view and percent of variance explained panel. The second model basis vector is reversed.

Simulated periodic data

In this section we analyze simulated data with two simplicity measures that are useful when we expect the data to have some periodic structure “hidden” in the nearly null space. The first measure is the built-in periodic measure. While this built-in measure may be fine on its own, it produces simplicity basis vectors that are rough. We have found that the periodic simplicity measure is most useful in combination with a divided difference measure, to force smoothness of basis functions. That is the subject of our second analysis, which demonstrates how to input a function for a user-defined simplicity measure.

For both examples we use simulated data generated from sine functions. Parts of this analysis can be seen in the R demo in `prinsimp` via the command `demo(periodic)`, which will also provide the user with the function `periodic.example`. We have fine-tuned the data parameters to make our point, but have left enough flexibility in our code to allow the interested user to experiment.

The data from the i th sampling unit, $i = 1, \dots, n$, are constructed from the process

$$y_i(x) = \alpha_{0,i} + \alpha_{1,i} \sin(2\pi x/L) + \alpha_{2,i} \sin(2\pi x/M) + \sum_{k=3}^{N+2} \alpha_{k,i} \sin(2\pi(k-2)x/K)$$

where $\alpha_{k,i}$ is distributed as $N(0, a_k^2)$, $k = 0, \dots, N + 2$. The data are $y_{ij} \equiv y_i(j) + \epsilon_{ij}$ with ϵ_{ij} distributed as $N(0, e^2)$ for $j = 1, \dots, J$. All random variables are independent.

The function `periodic.example` produces one simulated data set of n independent vectors, each of length $J = x.max$.

```
> periodic.example(L = 72, M = 5, K = 12, a0, a1, a2, sd.sigma, e, n = 100, x.max)
```

The arguments of `periodic.example` are L, M and K , which define the periods of the different components of the $N + 2$ underlying sine functions, and the standard deviations of the random variables: $a_0 = a_0, a_1 = a_1, a_2 = a_2$, `sd.sigma` equals the vector $(a_3, \dots, a_{N+2})^T$ and $e = e$. We simulate a $J = 72$ by $n = 100$ data matrix for analysis.

```
> # Simulate using demo(periodic)
> example <- periodic.example(a0 = 4, a1 = 4, a2 = 0, sd.sigma = 0.2,
+                             e = 1, x.max = 72)
```

Thus the data from subject i are obtained from the process

$$y_i(x) = \alpha_{0,i} + \alpha_{1,i} \sin(2\pi x/72) + \alpha_{3,i} \sin(2\pi x/12)$$

observed at $x = 1, \dots, 72$, with measurement error added, with error standard deviation 1. The standard deviations of $\alpha_{0,i}, \alpha_{1,i}$ and $\alpha_{3,i}$ are, respectively, 4, 4 and 0.2. Figure 4 contains a plot of the the first 15 data sets.

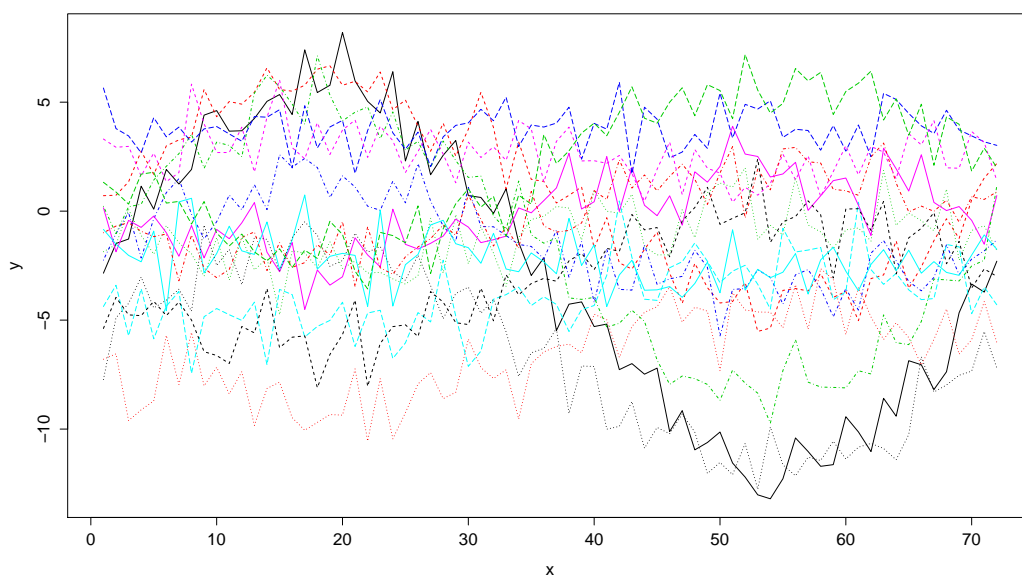


Figure 4: The plot of the data from the first 15 of the 100 curves of simulated periodic data observed at 72 values of x .

We will use `simpart` to analyze our high-dimensional data and see if we can locate any periodic structure. Recall that we simulated the data $\{y_{ij}\}$ using sine functions $\sin(2\pi \cdot /72)$ and $\sin(2\pi \cdot /12)$, functions whose periods are, respectively, 72 and 12, and that the variance of the random coefficient of the period 12 function is small. We show that PCA will not pick up the period 12 variability in the data, nor will `simpart` with the periodic simplicity measure. However, `simpart` with a smoothed periodic simplicity measure easily identifies the period 12 structure.

We begin with model space determination. This determination is the same as determining the number of components to retain in principal component analysis: by using either `princomp` or `simpart` with `simplifiedim = 0`, we can study the percent of variance explained. With this analysis, we see that a model space of dimension $J = 2$ explains approximately 96% of the sample variability, and the third principal component only explains an additional 0.22%. Figure 5 shows that the first principal component captures the constant function structure while the second principal component captures the structure with period 72. Moreover, we see no structure in the third through the sixth principal components. Figure 5 was produced as follows, using `basisplot`.

```
> # We do not need to specify the x vector because
> # the data are present in unit increments from 1 to 72, the default for x
> periodic.full <- simpart(example, simplifiedim = 0, "periodic", period = 12)
> par(mfrow = c(3, 2))
> basisplot(periodic.full, display = list(model = 1:6))
```

The chosen measure has no effect on the model basis vectors, as these vectors are simply the principal components of the covariance matrix.

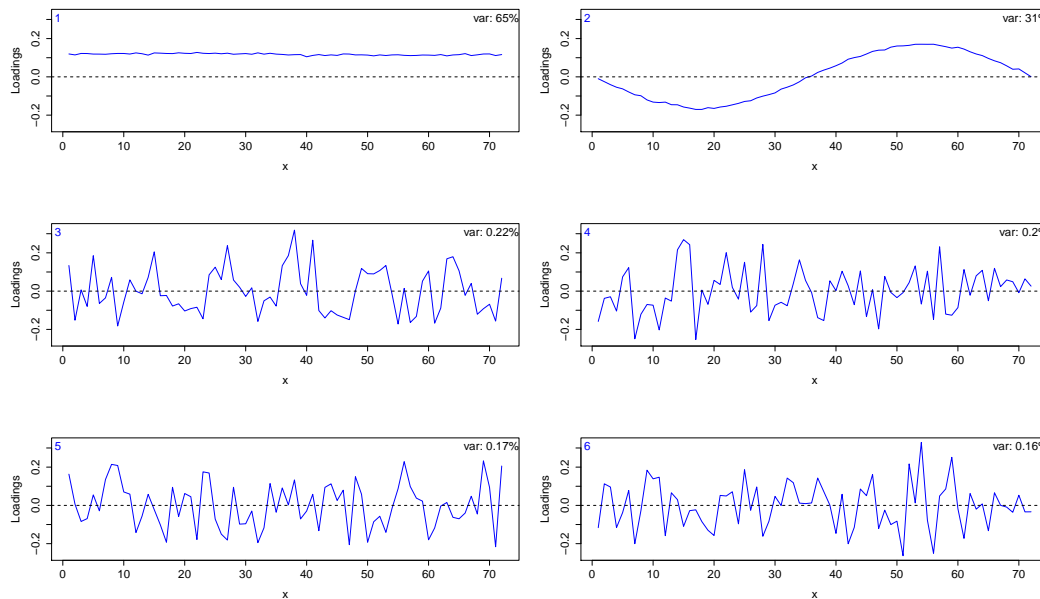


Figure 5: Analysis of the 100 curves of the simulated periodic data set (see Figure 4). Plotted are the first six principal components of the 72×72 sample covariance matrix, illustrating that PCA is unable to capture the period-12 component of variation.

Most researchers would see no reason to look beyond a 2-dimensional model space. But suppose we expect to find low variability structure with period of 12 in our data. We input our data to `simpart` specifying a periodic simplicity measure with period of 12. Since $x = (1, 2, \dots, 72)^T$, the index period and the usual time-based period are the same.

```
> periodic.sim <- simpart(example, simpledim = 70, measure = "periodic", period = 12)
> # Produce Figure 6.
> par(mfrow = c(3, 4)); basisplot(periodic.sim, display = list(simple = 1:12))
```

In Figure 6, we see that the first 12 vectors in the simplicity basis are quite rough and explain little of the variance. Our goal is to find simplicity basis vectors that are simple (periodic or close to periodic with period 12) and explain some non-negligible proportion of the variance. The easiest way to find these simplicity basis vectors is via the plot in Figure 7, produced by the following.

```
> plot(periodic.sim$variance$simple, periodic.sim$simplicity$simple,
+      xlab = "Variance", ylab = "Simplicity")
```

We see that there are no vectors that have both a high simplicity score and explain noticeably more variance than the others.

Plotting individuals' simplicity and pc scores

Just as in principal components analysis, we can look at individuals' scores to detect outliers or group structure. For a given basis vector v and an individual with mean-centered data vector w , the associated score is $v'w$. When v is a model space basis vector, $v'w$ is the usual principal components score. If v is a nearly null space basis vector, we say that $v'w$ is the simplicity score. Figure 8 shows the plot generated by the following command. The plot isn't very interesting, due to the regularity of our simulated data.

```
> plot(periodic.sim$scores[,1], periodic.sim$scores[,3],
+      xlab = "Score on first model basis vector",
+      ylab = "Score on first simplicity basis vector")
```

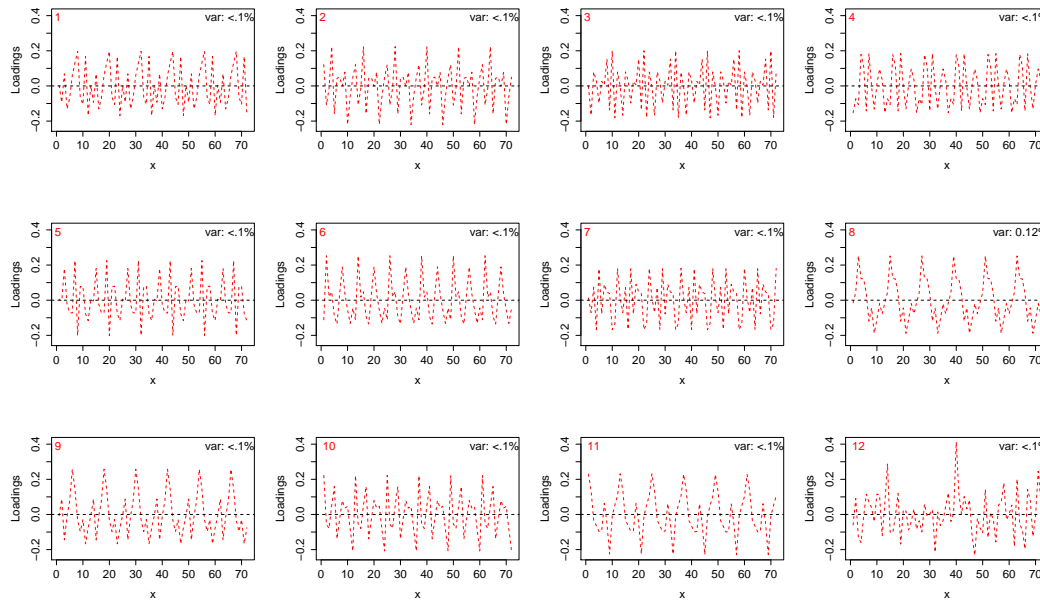


Figure 6: Prinsimp analysis of the simulated periodic data set. Plotted are the first 12 simplicity basis vectors using a model space of dimension 2 and periodic simplicity measure with period of 12. We see that this analysis is unable to capture the period-12 component of variation

Periodic example with user defined simplicity measure

In addition to `simpart`'s three built-in measures, `simpart` allows the user to pass a user-defined function to measure. We provide details for one function, which calculates a weighted combination of the two built-in measures, the second-divided difference measure and the periodic simplicity measure.

The previous `simpart` analysis used `measure = "periodic"`. However, the periodic simplicity measure was not able to pick up any structure in the nearly null space. In addition, the simplicity basis vectors were very rough. If we expect to see low variability structure in our data that is not only periodic but also smooth, we should incorporate smoothness into our periodic simplicity measure. An easy way to do that is by using a simplicity measure that is a weighted sum of the periodic and the second divided difference measures. This will force the leading periodic simplicity basis vectors to be smoother. We show how to define such a simplicity measure by writing the function `blend_measures` and passing it to `simpart`. If Λ_2 and Λ_π are the $d \times d$ non-negative definite matrices that define the second divided differences measure and periodic simplicity measure, respectively, then we combine them by defining our new simplicity measure based on $(1 - \alpha)\Lambda_2 + \alpha\Lambda_\pi$ for a user-chosen α in $[0, 1]$. The function `blend_measures` creates this matrix.

```
> blend_measures <- function(alpha, x, period) {
+   (1-alpha) * prinsimp:::lambda_second(x) +
+   (alpha) * prinsimp:::lambda_periodic(x, period)
+ }
> # Recall that the periodic simplicity measure requires an extra period argument
```

Notice that if we set `alpha = 1` then we have the built-in periodic simplicity measure, and `alpha = 0` gives us the second divided difference measure. Note that all user defined functions must include the argument `x`, but can also have additional arguments that are provided in the call to `simpart`, in this case `alpha` and `period`.

We analyze the same simulated data set as in the previous section, still using a two-dimensional model space. As before, we analyze our data with a period of 12 in the periodic simplicity measure and try to find important simplicity basis vectors by considering Figure 9.

```
> # We will use a 50-50 weighting of the two periodic and
> # second divided difference Lambda matrices
> periodic.blend <- simpart(example, simplifiedim = 70,
+   measure = "blend_measures", alpha = 0.5, period = 12)
> plot(periodic.blend$variance$simple, periodic.blend$simplicity$simple,
+   xlab = "Variance", ylab = "Simplicity Score")
```

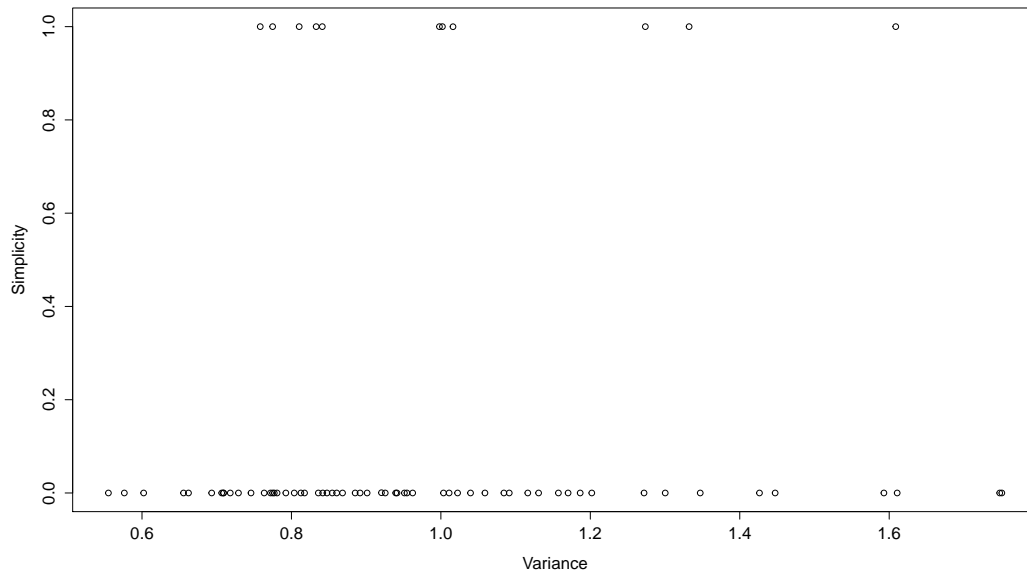


Figure 7: From the analysis as in Figure 6, a plot of the simplicity scores versus the associated variance explained for the 70 simplicity basis vectors. No simplicity basis vector stands out as “important” - as explaining a lot of the variation and having a high simplicity score.

The outlying point in the upper right corner is from the second simplicity basis vector. This vector corresponds to a variance of 2.20, which explains about 0.14% of the total variability. Clearly, the second simplicity basis vector captures variability caused by the sine function of period 12 (Figure 10). For this information and to create Figure 10, we used the following code.

```
> periodic.blend$variance$simple[2]
> periodic.blend$varperc$simple[2]
> basisplot(periodic.blend, display = list(simple = 2))
> lines(0.16 * sin(2*pi*(1:72)/12))
```

This analysis shows how to use the function `blend_measures` to locate an important and simple direction of variability, a direction that was not captured by principal component analysis. This example also demonstrates how the periodic measure is used: the user has some idea about the structure of data and so can propose a period. The user is encouraged to experiment with the measures.

A simple user defined simplicity measure

In some applications, we may want to consider that a vector is complex if the variance of its components is large - that is, if the components are very dissimilar. This might be sensible if, for instance, the data vectors' entries are different variables measured on similar scales, such as test scores. Thus, the complexity of the n -vector v is $\sum_j (v[j] - \bar{v})^2$ where \bar{v} is the average of the components of v . We can write this complexity as a quadratic form $v^T \Lambda v$: $\sum_j (v[j] - \bar{v})^2 = v^T v - \bar{v}^2/n = v^T (I - \mathbf{1}\mathbf{1}^T/n)v$ where $\mathbf{1}$ is an n -vector of ones. Since the maximum eigenvalue of $I - \mathbf{1}\mathbf{1}^T/n$ is 1, our simplicity measure uses the matrix $\Lambda = \mathbf{1}\mathbf{1}^T/n$. We can use this simplicity measure in `simpart` by defining the function `constant_simple`. Since our simplicity measure only depends on the length of the input vector, we simply input any x that is the same length as a data vector v .

```
> constant_simple <- function(x) {
  n <- length(x)
  matrix(1,n,n)/n
}
```

We use this function in `simpart` via `measure=constant_simple` as follows.

```
> constant.example <- simpart(caterpillar, simpledim = 4,
+                             measure = "constant_simple", x = x.cat)
```

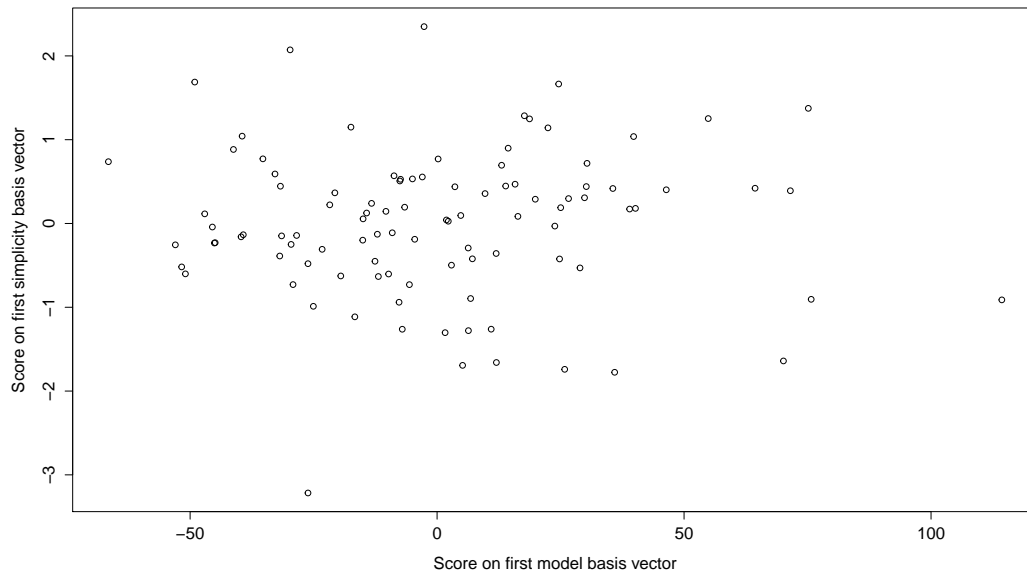


Figure 8: From the analysis as in Figures 6 and 7, a scatterplot of individuals' first principal components scores and first simplicity basis scores.

Implementation details

The core functionality of the **prinsimp** package is accessed through the `simpart` function. This function, as illustrated in examples above, takes in the data y in the form of a matrix or a formula object, the independent variable x , the dimension of the nearly null space (argument `simplifiedim`), and the simplicity measure (argument `measure`). Optionally, y can contain the covariance matrix itself, which is signalled with argument `cov=TRUE`.

The value returned by `simpart` is an S3 object of class "simpart", with the following named components:

- `model, simple`: basis of the model and nearly null space, respectively, with vectors arranged in columns.
- `variance`: a list of variances associated with the vectors in the model and nearly null spaces (components `model` and `simple`), as well as the eigenvalues of the covariance matrix (component `full`).
- `varperc`: a list of the percent of variance explained by the corresponding basis vector in the model and nearly null space (components `model` and `simple`).
- `simplicity`: a list of simplicity values of vectors in the model and simplicity basis (components `model` and `simple`), and the simplicity values of the simplicity basis when `simplifiedim=d` (component `full`).
- `measure, call`: the simplicity measure used and the call object

In addition, when the y argument is a data rather than covariance matrix, the result includes the component scores, for scores on the basis vector loadings.

The **prinsimp** package provides a range of methods to view objects of class `simpart`. In addition to the `print` and summary methods for printing out the basis vectors, simplicity scores, and variances to the console, the user can also view them graphically. The `plot` method that we have used above puts together the collage of basis vectors along with variance-simplicity view and the percent of variance explained panel. The individual subplots can also be displayed alone, using the functions `basisplot`, `varsimp`, and `varperc`. (The `plot` method merely sets up the layout and calls those functions for the actual plotting.)

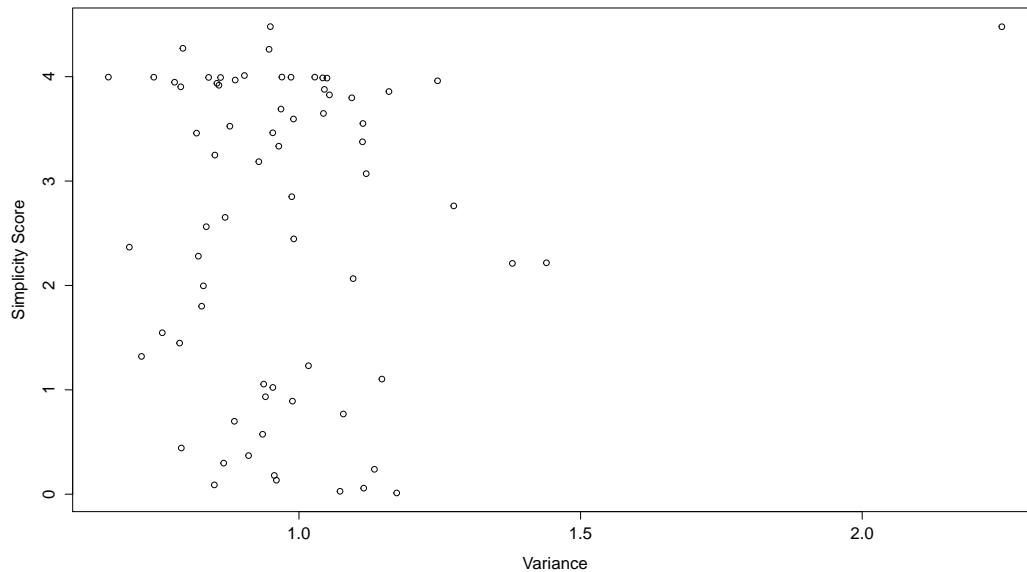


Figure 9: The simplicity score versus the variance explained of the 70 simplicity basis vectors in the analysis of the periodic data with the blended periodic-smooth simplicity measure and period 12. The outlying point in the upper right corner indicates an “important” simplicity basis vector - one that explains a lot of the variability and has a high simplicity score. This outlying point corresponds to the second simplicity basis vector, shown in the next figure.

Writing custom simplicity functions

As mentioned earlier, as an alternative to using one of the built-in simplicity measures, the user can provide his or her own measure by passing a custom function as the value of the measure argument. This function returns the Λ matrix of the measure; i.e., the simplicity score of a vector $v \in \mathcal{R}^d$ is $v^T \Lambda v$. The custom function has to have at least one argument, named x , that receives values of the independent variable x as given to `simplart`. The custom function is allowed to have additional arguments; these are included in the call to `simplart`, and simply passed through to the measure function.

Bibliography

- D. Cubranic, J. Zhang, N. Heckman, T. Gaydos, , and J. Marron. *prinsimp: Finding and plotting simple basis vectors for multivariate data*, 2013. URL <http://CRAN.R-project.org/package=prinsimp>. R package version 0.8-8. [p27]
- E. Demidenko. *Mixed Models: Theory and Applications*. Wiley Series in Probability and Statistics. Wiley-Interscience, Hoboken, NJ, 2004. [p30]
- P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, 1996. [p28]
- T. Gaydos, N. E. Heckman, M. Kirkpatrick, J. Stinchcombe, J. Schmitt, J. Kingsolver, and J. S. Marron. Visualizing genetic constraints. *Annals of Applied Statistics*, 7(2):860–882, 2013. [p27, 28, 30, 33]
- P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: a Roughness Penalty Approach*. Monographs on Statistics and Applied Probability. Chapman & Hall, London, 1994. [p28]
- R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Pearson, 2007. [p27]
- J. G. Kingsolver, G. J. Ragland, and J. G. Shlichta. Quantitative genetics of continuous reaction norms: thermal sensitivity of caterpillar growth rates. *Evolution*, 58:1521–1529, 2004. [p30]
- M. Schatzman. *Numerical Analysis: A Mathematical Introduction*. Clarendon Press, Oxford, 2002. [p28]

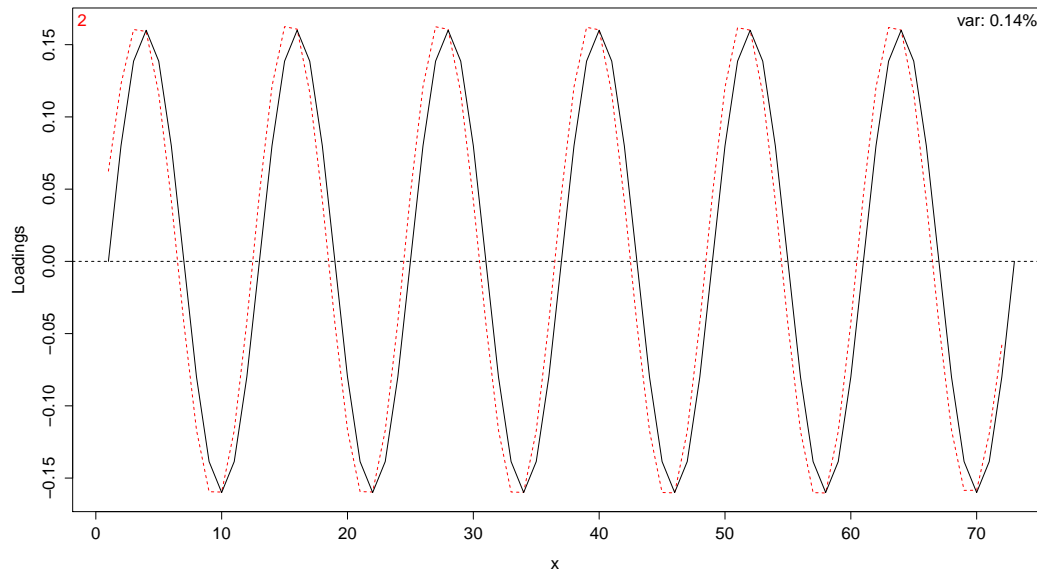


Figure 10: The second simplicity basis vector (red dashed line), which corresponds to the outlying point in the previous figure. Overlaid is a scaled sine function with period 12 (black line), indicating that the **prinsimp** analysis has captured the period-12 variability.

F. Yao, H.-G. Mueller, and J.-L. Wang. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100:577–590, 2005. [p30]

X. Zhao, J. S. Marron, and M. T. Wells. The functional data analysis view of longitudinal data. *Statistica Sinica*, 14:789–808, 2004. [p30]

Jonathan Zhang
 Statistics Department, University of British Columbia
 Vancouver BC
 Canada
jono_722@hotmail.com

Nancy Heckman
 Statistics Department, University of British Columbia
 Vancouver BC
 Canada
nancy@stat.ubc.ca

Davor Cubranic
 Statistics Department, University of British Columbia
 Vancouver BC
 Canada
cubranic@stat.ubc.ca

Joel G. Kingsolver
 Department of Biology
 University of North Carolina,
 Chapel Hill, NC
jgking@bio.unc.edu

J.S. Marron
 Statistics Department
 University of North Carolina,
 Chapel Hill NC

marron@unc.edu

Travis L. Gaydos
The MITRE Corporation
McLean, VA
travis.gaydos@gmail.com

phaseR: An R Package for Phase Plane Analysis of Autonomous ODE Systems

by Michael J. Grayling

Abstract When modelling physical systems, analysts will frequently be confronted by differential equations which cannot be solved analytically. In this instance, numerical integration will usually be the only way forward. However, for autonomous systems of ordinary differential equations (ODEs) in one or two dimensions, it is possible to employ an instructive qualitative analysis foregoing this requirement, using so-called phase plane methods. Moreover, this qualitative analysis can even prove to be highly useful for systems that can be solved analytically, or will be solved numerically anyway. The package **phaseR** allows the user to perform such phase plane analyses: determining the stability of any equilibrium points easily, and producing informative plots.

Introduction

Repeatedly, when a system of differential equations is written down, it cannot be solved analytically. This is particularly true in the non-linear case, which unfortunately habitually arises when modelling physical systems. As such, it is common that numerical integration is the only way for a modeller to analyse the properties of their system. Consequently, many software packages exist today to assist in this step. In R, for example, the package **deSolve** (Soetaert et al., 2010) deals with many classes of differential equation. It allows users to solve first-order stiff and non-stiff initial value problem ODEs, as well as stiff and non-stiff delay differential equations (DDEs), and differential algebraic equations (DAEs) up to index 3. Moreover, it can tackle partial differential equations (PDEs) with the assistance **ReacTran** (Soetaert and Meysman, 2012). For such PDE problems users also have available the package **rootSolve** (Soetaert et al., 2009), which is additionally particularly suited to analysing the equilibrium of differential equations. Finally, **bvpSolve** (Soetaert et al., 2013) can tackle boundary value problems of systems of ODEs, whilst **sde** (Iacus, 2009) is available for stochastic differential equations (SDEs). However, for autonomous ODE systems in either one or two dimensions, phase plane methods, as developed by mathematicians such as Henri Poincaré in the 19th Century (Farlow, 2006), allow for a qualitative analysis of the system's properties without the need for numerical integration. Specifically, it is usually possible to determine the long-term behaviour of the system for any initial condition, via a highly graphical procedure coupled with algorithmic mathematical analysis.

As a result of these considerations, phase plane analysis is an important device in any modeller's toolbox. It stands as a key technique not only for mathematicians, but biologists, physicists and engineers amongst others. See for example Jordan and Smith (2007), Barnes and Fulford (2009), Choudhury (2005), and Mooney (1999) for descriptions of uses in various fields. Since the method is extremely algorithmic in nature, software tools would clearly provide a powerful system by which to execute the techniques. Despite this, few programmes are available for implementing phase plane methods. Presently, a programme entitled *pplane* that employs a simple GUI system is available for Matlab (MATLAB, 2014; Polking, 2009). In addition, several java applets are available across the internet for executing phase plane methods. But, they vary widely in quality and often struggle when confronted with complex systems, whilst they also provide no simple way to export produced plots. Now, with R ever growing in popularity as an open-source general purpose scientific computing language, it is logical that available code to implement phase plane methods would serve as a valuable resource. The first attempts in this direction were implemented by Kaplan and Flath (2004) at Macalester College who created several short programmes to analyse two dimensional ODEs and provided several example systems. **phaseR** extends their initial ideas in order to create a complete package; one providing phase plane novices with the opportunity for independent learning, instructors with a useful package for practical sessions, and experienced modellers with an easy means by which to produce quality plots.

Overall, **phaseR** maintains the original structure of Kaplan and Flath's code; with key functions available for each of the steps of a full phase plane analysis. Specifically, it can assist in identifying and classifying equilibrium points, allows direction fields as well as nullclines to be plotted, and allows trajectories to be added to plots for user specified initial conditions. **phaseR**, however, extends the functionality of the code to allow analysis of both one and two dimensional systems, whilst also allowing for far greater control over the final produced plot. Moreover, **phaseR** comes complete with an extensive guide detailing the techniques of phase plane analysis, also containing numerous examples and exercises. Finally, **phaseR** makes use of available R packages for finding numerical solutions to ODEs in order to ensure maximum stability of the required integration step. I will now proceed by briefly reviewing the techniques of phase plane analysis, before discussing the usage of the

package through several examples.

One dimensional autonomous ODE systems

Any one dimensional autonomous ODE, of a function $y = y(t) = y_t$, can be written in the following form

$$\frac{dy_t}{dt} = f(y_t).$$

Thus, autonomous systems are characterised by not explicitly depending on the independent variable t . For such systems, phase plane analysis begins by plotting at a range of values for both the dependent and the independent variable, a small arrow indicating the rate of change of y_t as provided by the ODE. This plot, commonly referred to as the direction field, is useful because solutions to the ODE must pass through the arrows in a tangential manner. Therefore, once produced, it provides an easy construction from which to draw trajectories for any initial condition without the need to solve the ODE.

Following this, so-called equilibrium points are determined. Defined as the points y_* where $f(y_*) = 0$, the reason for their name is easy to see: beginning at such a point, because of the lack of explicit dependence upon t in f , means the solution will remain there for all values of t . Moreover, these points are then classified as either stable or unstable, depending upon whether solutions converge towards, or diverge away, from them. Stability may here be determined from a 'phase portrait plot' of $f(y_t)$ against y_t ; on which arrows are placed indicating the direction of change of y_t with t . Arrows pointing towards each other on either side of an equilibrium point denote stability, whilst arrows pointing away from each other indicate the presence of an unstable point. Alternatively, a Taylor Series argument can be utilised to define a simple criterion. The argument proceeds by supposing the system begins a small distance δ_0 away from the fixed point y_* , i. e. $y_0 = y_* + \delta_0$. Then, writing our general expression for y_t as $y_t = y_* + \delta_t$, we use the Taylor Series expansion of f to form a differential equation for how δ_t changes with t

$$f(y_* + \delta_t) = f(y_*) + \delta_t \frac{\partial f}{\partial y_t}(y_*) + o(\delta_t),$$

where we have assumed higher order terms are negligible. Recalling $f(y_*) = 0$, our ODE becomes

$$\begin{aligned} \frac{d}{dt}f(y_* + \delta_t) &= \delta_t \frac{\partial f}{\partial y_t}(y_*), \\ \Rightarrow \frac{d\delta_t}{dt} &= \delta_t \frac{\partial f}{\partial y_t}(y_*) = k\delta_t. \end{aligned}$$

This autonomous ODE for δ_t can be solved easily to give $\delta_t = \delta_0 e^{kt}$. This analysis is useful to us since stability can be determined based upon whether δ_t grows, or decays, as t increases, i. e. using the simple criterion

$$k = \frac{\partial f}{\partial y_t}(y_*) = \begin{cases} > 0 & \text{Stable,} \\ < 0 & \text{Unstable.} \end{cases}$$

Two dimensional autonomous ODE systems

In the two dimensional case, for functions $x = x(t) = x_t$ and $y = y(t) = y_t$, any autonomous ODE system can be written as

$$\frac{dx_t}{dt} = f(x_t, y_t), \quad \frac{dy_t}{dt} = g(x_t, y_t).$$

Thus as before, these systems are characterised by a lack of explicit dependence upon the independent variable t in the functional forms of f and g .

The direction field, more commonly referred to here as the velocity field, is again produced to provide a powerful way by which to plot trajectories for any initial condition. However, here this is done so in the x_t - y_t plane, rather than that containing the independent variable. This visualisation proves to be the best way by which to examine an autonomous two dimensional system.

Commonly, to assist the plotting of the velocity field, nullclines are first computed. These are defined as the location where $f(x_t, y_t) = 0$ or $g(x_t, y_t) = 0$. Thus they define the curves across which either x_t or y_t does not change in t . Any velocity arrows produced will therefore either be perfectly horizontal or vertical. Since a velocity field must vary continuously across a particular nullcline, by the uniqueness of solutions to ODEs, nullclines can be used to check that no arrows have been plotted

incorrectly.

As one would expect, equilibrium points retain their great importance in two dimensions. Here, their definition generalises as the locations (x_*, y_*) where $f(x_*, y_*) = g(x_*, y_*) = 0$. Consequently, the intersection of the x_t and y_t nullclines can be used to determine their location. Stability here is defined in an analogous manner to the one dimensional case, whilst a Taylor Series argument again allows the formulation of a simple rule for determining said stability. As before, it supposes that we have an equilibrium point, now given by (x_*, y_*) , and that the system initially lies slightly away from this point at $(x_* + \delta_0, y_* + \epsilon_0)$, and in general we have $(x_t, y_t) = (x_* + \delta_t, y_* + \epsilon_t)$. Then using the Taylor expansion for f , the differential equation for x_t becomes

$$\begin{aligned} \frac{d\delta_t}{dt} &= f(x_* + \delta_t, y_* + \epsilon_t), \\ &= f(x_*, y_*) + \delta_t \frac{\partial f}{\partial x_t}(x_*, y_*) + \epsilon_t \frac{\partial f}{\partial y_t}(x_*, y_*) + o(\delta_t) + o(\epsilon_t), \\ &= \delta_t \frac{\partial f}{\partial x_t}(x_*, y_*) + \epsilon_t \frac{\partial f}{\partial y_t}(x_*, y_*) + o(\delta_t) + o(\epsilon_t). \end{aligned}$$

Similarly, the differential equation for y_t becomes

$$\frac{d\epsilon_t}{dt} = \delta_t \frac{\partial g}{\partial x_t}(x_*, y_*) + \epsilon_t \frac{\partial g}{\partial y_t}(x_*, y_*) + o(\delta_t) + o(\epsilon_t).$$

In both equations we have again assumed terms of second order and higher are negligible. Now if we write this system in matrix form we acquire

$$\frac{d\delta_t}{dt} = \begin{pmatrix} \frac{\partial f}{\partial x_t} & \frac{\partial f}{\partial y_t} \\ \frac{\partial g}{\partial x_t} & \frac{\partial g}{\partial y_t} \end{pmatrix} \Big|_{(x_*, y_*)} \delta_t = J\delta_t.$$

Here J is called the Jacobian of the system, and $\delta = (\delta, \epsilon)^\top$. It is consideration of the characteristic equation of J , and the values of its eigenvalues it provides, that allows equilibrium points to be classified, not only as either stable or unstable, but also in to sub-categories. Full details of this can be found in the guide that accompanies **phaseR**, available in the documentation folder of its installation. However, classification ultimately, and simply, depends upon the values of the determinant and trace of J .

Thus, in both the one and two dimensional cases, phase plane methods allow trajectories to be easily plotted, and the long-term behaviour of a system identified by classifying the equilibria. We shall next see how **phaseR** can be used to perform such analyses using several simple commands.

Key functions

phaseR employs six key functions for the employment of phase plane analysis.

- **flowField**: Plots the direction or velocity field of one or two dimensional autonomous ODE systems.
- **nullclines**: Plots the nullclines of two dimensional autonomous ODE systems. Alternatively, it can be used to plot horizontal lines at equilibrium points for one dimensional autonomous ODE systems.
- **numericalSolution**: For two dimensional systems, this function numerically solves the ODEs for a given initial condition via **deSolve**, and then plots the dependent variables against the independent variable. Thus, it behaves as a wrapper for the user to **deSolve**, allowing for easier implementation and subsequent plotting.
- **phasePortrait**: For one dimensional autonomous ODEs, it plots the phase portrait i. e. the derivative against the dependent variable. In addition, it adds arrows to the dependent variable axis from which the stability of any equilibrium points can be determined.
- **stability**: Classifies equilibrium points, using the criteria that result from the aforementioned Taylor Series arguments.
- **trajectory**: Plots trajectories in the x_t - y_t plane by performing numerical integration of the chosen ODE system, again via **deSolve**. Initial conditions can be specified as an argument, or by clicking with the cursor on locations in a pre-existing plot.

As an example, standard phase plane analyses for a two dimensional ODE system would proceed by using `flowField`, `nullclines` and then `trajectory` to create a summarising plot, and finally stability to classify the equilibria.

Example 1: logistic growth model

To describe how **phaseR** can be used to analyse a one dimensional autonomous system of interest, we will consider the logistic growth model. Originally proposed by [Verhulst \(1838\)](#), this ODE is frequently used in Biology to model the growth of a population under density dependence. It is given in its most general case by

$$\frac{dy_t}{dt} = \beta y_t \left(1 - \frac{y_t}{K}\right).$$

To analyse any system using **phaseR**, a function must first be created to return the value of the derivative at any point (t, y) , in a style compatible with **deSolve**. For this model such a function is already present in the package as `logistic`, which comprises the following code

```
logistic <- function(t, y, parameters){
  beta <- parameters[1]
  K <- parameters[2]
  dy <- beta*y*(1 - y/K)
  list(dy)
}
```

The general format of this derivative function should be a common one: given t , y and a vector `parameters`, it should simply return a list with first element the value of the derivative.

With this, we can then proceed with our desired analysis. From here we shall consider the particular case $\beta = 1$ and $K = 2$. The following code creates [Figure 1](#); adding the direction field, several trajectories and horizontal lines at any equilibrium points.

```
> logistic.flowField <-
+   flowField(logistic, x.lim = c(0, 5), y.lim = c(-1, 3),
+             parameters = c(1, 2), points = 21,
+             system = "one.dim", add = FALSE, xlab = "t")
> grid()
> logistic.nullclines <-
+   nullclines(logistic, x.lim = c(0, 5), y.lim = c(-1, 3),
+             parameters = c(1, 2), system = "one.dim")
> logistic.trajectory <-
+   trajectory(logistic, y0 = c(-0.5, 0.5, 1.5, 2.5), t.end = 5,
+             parameters = c(1, 2), system = "one.dim", colour = rep("black", 4))
```

Here we have tailored our result by setting a suitable range for the dependent and independent variables, whilst by default suitable axes labels have been added to the plot. These are just several input variables that can be changed to suit the needs of your particular system.

Now, we can observe from [Figure 1](#) that two equilibrium points have been identified; appearing to be $y_* = 0$ and $y_* = 2$. We can confirm their location analytically however by setting the right hand side of the ODE to zero

$$y_* \left(1 - \frac{y_*}{2}\right) = 0, \\ \Rightarrow y_* = \{0, 2\}.$$

It appears from the trajectories in [Figure 1](#) that the point $y_* = 2$ is stable, and $y_* = 0$ unstable. We can confirm this by plotting the phase portrait with the following code, producing [Figure 2](#)

```
> logistic.phasePortrait <- phasePortrait(logistic, y.lim = c(-0.5, 2.5),
+                                       parameters = c(1, 2), points = 10)
> grid()
```

Alternatively, if we use our Taylor Series method we can draw this same conclusion

$$\left. \frac{d}{dy} \left(\frac{dy}{dt} \right) \right|_{y=y_*} = 1 - y_* = \begin{cases} +1 & y_* = 0, \\ -1 & y_* = 2 \end{cases}.$$

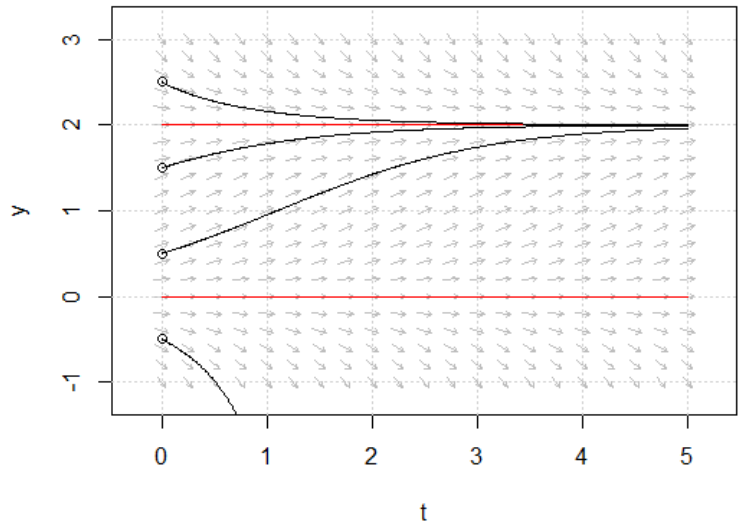


Figure 1: The direction field and several trajectories for the logistic growth model with $\beta = 1$ and $K = 2$. It can be seen that two equilibrium points have been located.

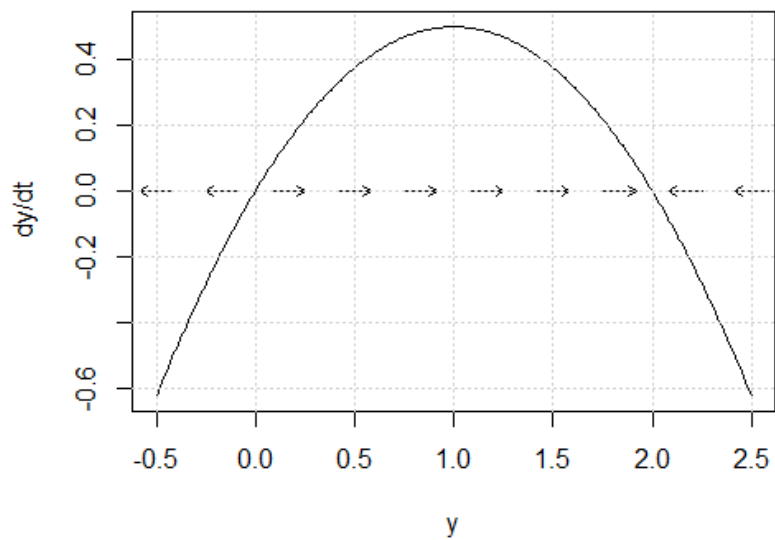


Figure 2: The phase portrait for the logistic growth model with $\beta = 1$ and $K = 2$. The line $y = 0$ has been identified as unstable, and the line $y = 2$ as stable.

Thus we have drawn the same conclusion; $y_* = 0$ is indeed unstable, and $y_* = 2$ stable. The following code can perform this analysis for us using `stability`

```
> logistic.stability.1 <-
+   stability(logistic, y.star = 0, parameters = c(1, 2), system = "one.dim")
```

Discriminant: 1 Classification: Unstable

```
> logistic.stability.2 <-
+   stability(logistic, y.star = 2, parameters = c(1, 2), system = "one.dim")
```

Discriminant: -1 Classification: Stable

This completes our phase plane analysis of the logistic growth model. We have successfully identified that for this model, if $y_0 > 0$, which it must be for the model to make physical sense, the system will eventually approach the line $y_t = 2$, regardless of the exact initial condition. **phaseR** clearly provided a capacity by which to either check, or perform, the analyses that would take time by hand using several short lines of code.

Example 2: Lotka-Volterra model

As an example of the capabilities of **phaseR** for a two dimensional system, we will study a particular case of the Lotka-Volterra model (Lotka, 1925; Volterra, 1926) for interacting predator and prey. This model can be written in its general form as

$$\begin{aligned}\frac{dx_t}{dt} &= \lambda x_t - \epsilon x_t y_t, \\ \frac{dy_t}{dt} &= \eta x_t y_t - \delta y_t.\end{aligned}$$

Again, we must first specify a derivative function. As before, this is already provided in the package as `lotkaVolterra`, which consists of the following code

```
lotkaVolterra <- function(t, y, parameters) {
  x <- y[1]
  y <- y[2]
  lambda <- parameters[1]
  epsilon <- parameters[2]
  eta <- parameters[3]
  delta <- parameters[4]
  dy <- numeric(2)
  dy[1] <- lambda*x - epsilon*x*y
  dy[2] <- eta*x*y - delta*y
  list(dy)
}
```

deSolve again requires us here to create a function taking values for t , y and $parameters$, but here returning a list whose first element is a vector of the two derivatives, and also accepting y as a vector argument.

From here we shall focus on the particular case $\lambda = 2$, $\epsilon = 1$, $\eta = 3$ and $\delta = 2$. We begin by determining the nullclines; by setting the two derivatives to zero in turn, starting with x_t

$$2x_t - x_t y_t = 0 \Rightarrow x_t(2 - y_t) = 0 \Rightarrow x_t = 0, y_t = 2,$$

and then y_t

$$3x_t y_t - 2y_t = 0 \Rightarrow y_t(3x_t - 2) = 0 \Rightarrow y_t = 0, x_t = 2/3.$$

Requiring that both derivatives are zero identifies the equilibrium points of the model; clearly given here by $(0, 0)$ and $(2/3, 2)$. To classify them we turn to the Jacobian

$$J = \begin{pmatrix} 2 - y_* & -x_* \\ 3y_* & 3x_* - 2 \end{pmatrix}.$$

Thus for $(0, 0)$; $\det(J) = -4$, which classifies the point as a saddle. Additionally, for $(2/3, 2)$ we have that $\det(J) = 4$, and $\text{tr}(J) = 0$, which classifies the point as a centre. Readers are again pointed to the package guide for further details of classification using the Jacobian.

We can repeat all of the above analyses easily using **phaseR**; plotting the nullclines along with the velocity field and several trajectories with the following code that produces Figure 3

```
> lotkaVolterra.flowField <-
+   flowField(lotkaVolterra, x.lim = c(0, 5), y.lim = c(0, 10),
+             parameters = c(2, 1, 3, 2), points = 19, add = FALSE)
> grid()
> lotkaVolterra.nullclines <-
+   nullclines(lotkaVolterra, x.lim = c(-1, 5), y.lim = c(-1, 10),
+             parameters = c(2, 1, 3, 2), points = 500)
> y0 <- matrix(c(1, 2, 2, 2, 3, 4), ncol = 2, nrow = 3, byrow = TRUE)
> lotkaVolterra.trajectory <-
+   trajectory(lotkaVolterra, y0 = y0, t.end = 10,
+             parameters = c(2, 1, 3, 2), colour = rep("black", 3))
```

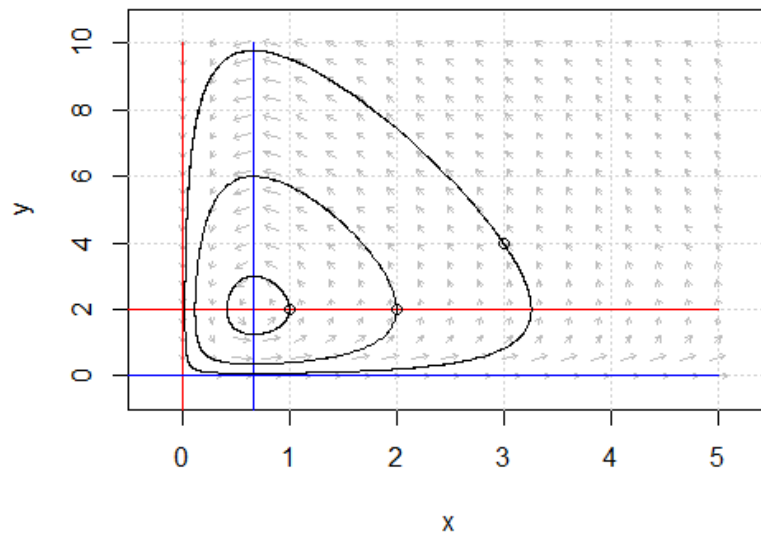


Figure 3: The velocity field, nullclines and several trajectories for the Lotka-Volterra model with $\lambda = 2$, $\epsilon = 1$, $\eta = 3$ and $\delta = 2$. Trajectories can be seen to traverse around the point $(2/3, 2)$.

Finally, we can verify that our stability analysis was indeed correct

```
> lotkaVolterra.stability.1 <-
+   stability(lotkaVolterra, y.star = c(0, 0), parameters = c(2, 1, 3, 2))

T: 0   Delta: -4   Discriminant: 16   Classification: Saddle

> lotkaVolterra.stability.2 <-
+   stability(lotkaVolterra, y.star = c(2/3, 2), parameters = c(2, 1, 3, 2))

T: 0   Delta: 4   Discriminant: -16   Classification: Centre
```

Biologically, we have seen that no matter what the exact initial starting values are, according to this model the numbers of predator and prey will oscillate around the centre $(2/3, 2)$. Alternatively, this oscillation can be seen by plotting the dependent variables against the independent using the following code, producing Figure 4

```
> lotkaVolterra.numericalSolution <-
+   numericalSolution(lotkaVolterra, y0 = c(3, 4), t.end = 10, type = "one",
+                   parameters = c(2, 1, 3, 2), colour = c("green", "orange"),
+                   ylab = "x, y", ylim = c(0, 10))
> legend("bottomright", col = c("green", "orange"), legend = c("x", "y"), lty = 1)
```

This concludes our analysis of this two dimensional system. Again **phaseR** allowed for the phase plane analysis to be performed quickly and easily, bypassing the time consuming process of drawing by hand.

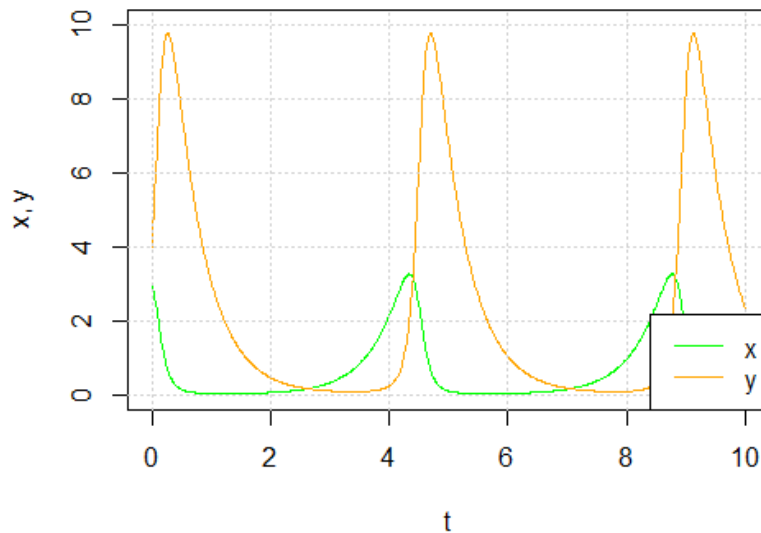


Figure 4: The numerical solution of the Lotka-Volterra model with $\lambda = 2$, $\epsilon = 1$, $\eta = 3$ and $\delta = 2$ and $(x_0, y_0) = (3, 4)$. The number of each species can be seen to oscillate.

Concluding remarks and future developments

Rarely can a system of differential equations be solved in closed form. Consequently, numerical solutions are often required and therefore much theory exists around integration techniques for each class of differential equations. In R many of these methods have been implemented in packages such as **deSolve**, **bvpSolve**, **sde**, **ReacTran** and **rootSolve**. Between them, these packages allow modellers to deal with many classes of ODEs, DDEs, DAEs, SDEs and PDEs. However, for a one or two dimensional system of autonomous ODEs, an alternative approach to direct numerical solution is available. This so-called phase plane analysis allows much important inference to be made about the system using relatively simplistic graphical and mathematical techniques. The algorithmic nature of the phase plane approach lends itself well to software tools, and **phaseR** provides the first such complete package in R for its implementation. I have demonstrated how **phaseR** allows users to easily perform this qualitative analysis with little initial system set-up, and few commands for execution, required. Sometimes, phase plane analysis is criticised on two grounds; that it is limited to systems of two or less dimension (Mutambara, 1999), and that it is highly labour intensive. Whilst the former still stands, and thus only problems that can be at least well approximated by a second order system may be analysed using the methods discussed above, the latter issue is at least relieved further by the creation of **phaseR**. Whilst to provide greater detail on the dimensional limitation future development of the package will seek to incorporate further example systems from nature, including approximation techniques for higher order problems. In addition, package development will see the one dimensional tools extended to non-autonomous ODEs. Therefore, given all these considerations, in conjunction with the extensive accompanying guide describing in increased detail both the above techniques and providing many worked examples and exercises, **phaseR** should hopefully serve as a useful package for both independent and group led learning, as well as for those simply seeking to produce high-quality figures.

Bibliography

- B. Barnes and G. R. Fulford. *Mathematical Modelling with Case Studies: A Differential Equations Approach Using Maple and MATLAB*, pages 159–258. CRC Press, Florida, United States, second edition, 2009. ISBN 978-1420083484. [p43]
- D. R. Choudhury. *Modern Control Engineering*, page 755. Prentice-Hall, New Delhi, India, 2005. ISBN 978-81-203-2196-0. [p43]
- S. J. Farlow. *An Introduction to Differential Equations and Their Applications*, page 452. Dover Publications, 2006. ISBN 978-0486445953. [p43]
- S. M. Iacus. *sde: Simulation and Inference for Stochastic Differential Equations*, 2009. URL <http://CRAN.R-project.org/package=sde>. R package version 2.0.10. [p43]

- D. W. Jordan and P. Smith. *Nonlinear Ordinary Differential Equations: An Introduction for Scientists and Engineers*, pages 49–88. Oxford University Press, New York, United States, fourth edition, 2007. ISBN 978-0-19-920824. [p43]
- D. Kaplan and D. Flath. Visualizing the Phase Plane using R, 2004. URL <http://www.macalester.edu/~kaplan/math135/pplane.pdf>. [p43]
- A. J. Lotka. *Elements of Physical Biology*. Williams and Wilkins, Baltimore, United States, 1925. [p48]
- MATLAB. 2014a. The MathWorks Inc., Natick, Massachusetts, United States, 2014. [p43]
- D. Mooney. *A Course in Mathematical Modeling*, page 283. The Mathematical Association of America, 1999. ISBN 978-0-88385-7120. [p43]
- A. G. O. Mutambara. *Design and Analysis of Control Systems*, page 722. CRC Press, 1999. ISBN 978-0849318986. [p50]
- J. C. Polking. pplane8, 2009. URL <http://math.rice.edu/~dfield/>. [p43]
- K. Soetaert and F. Meysman. Reactive transport in aquatic ecosystems: Rapid model prototyping in the open source software R. *Environmental Modelling & Software*, 32:49–60, 2012. [p43]
- K. Soetaert, T. Petzoldt, and R. W. Setzer. *A Practical Guide to Ecological Modelling. Using R as a Simulation Platform*. Springer, 2009. ISBN 978-1402086236. [p43]
- K. Soetaert, T. Petzoldt, and R. W. Setzer. Solving Differential Equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010. [p43]
- K. Soetaert, J. Cash, and F. Mazzia. *bvpSolve: Solvers for boundary value problems of ordinary differential equations*, 2013. URL <http://CRAN.R-project.org/package=bvpSolve>. R package version 1.2.4. [p43]
- P.-F. Verhulst. Notice sur la loi que la population poursuit dans son accroissement. *Correspondance mathématique et physique*, 10:113–121, 1838. [p46]
- V. Volterra. Variazioni e fluttuazioni del numero d’individui in specie animali conviventi. *Mem Acad Lincei Roma*, 2:31–113, 1926. [p48]

Michael J. Grayling
MRC Biostatistics Unit
Cambridge
CB2 0SR
United Kingdom
mjg211@cam.ac.uk

Flexible R Functions for Processing Accelerometer Data, with Emphasis on NHANES 2003–2006

by *Dane R. Van Domelen and W. Stephen Pittard*

Abstract Accelerometers are a valuable tool for measuring physical activity (PA) in epidemiological studies. However, considerable processing is needed to convert time-series accelerometer data into meaningful variables for statistical analysis. This article describes two recently developed R packages for processing accelerometer data. The package **accelerometry** contains functions for performing various data processing procedures, such as identifying periods of non-wear time and bouts of activity. The functions are flexible, computationally efficient, and compatible with uniaxial or triaxial data. The package **nhanesaccel** is specifically for processing data from the National Health and Nutrition Examination Survey (NHANES), years 2003–2006. Its primary function generates measures of PA volume, intensity, frequency, and patterns according to user-specified data processing methods. This function can process the NHANES 2003–2006 dataset in under one minute, which is a drastic improvement over existing software. This article highlights important features of packages **accelerometry** and **nhanesaccel** and demonstrates typical usage for PA researchers.

Introduction

The National Health and Nutrition Examination Survey (NHANES) is one of a select few national studies to include objective physical activity (PA) monitoring with accelerometers (Troiano et al., 2008; Hagstromer et al., 2007; Colley et al., 2011). In the 2003–2004 and 2005–2006 waves of the study, a total of 14,631 Americans age ≥ 6 years wore uniaxial ActiGraph GT1M accelerometers at the hip for seven consecutive days. The result is a rich dataset for describing PA levels in Americans and assessing correlates of PA.

In 2007, SAS programs for processing the accelerometer data from NHANES were made available on the National Cancer Institute's (NCI) website (National Cancer Institute, 2007). Using these programs, researchers around the world could access the time-series accelerometer data from NHANES and run SAS programs to derive meaningful PA variables to use in statistical analyses. Over the past seven years, the NCI's SAS programs have facilitated great progress in understanding the distribution of PA behaviors in America (Troiano et al., 2008; Tudor-Locke et al., 2010), and have helped to identify numerous cross-sectional associations between PA and health outcomes (Healy et al., 2011; Sisson et al., 2010; Carson and Janssen, 2011).

While the NCI's SAS programs provide several indicators of overall PA and moderate-to-vigorous PA (MVPA), researchers are increasingly writing customized scripts in programs such as MATLAB (The MathWorks, Inc., 2014), LabVIEW (National Instruments, 2014), and R for more flexibility in data processing (Tudor-Locke et al., 2011). This approach allows researchers to generate more descriptive variables, such as time spent in extended periods of sedentariness or hour-by-hour count averages (Bankoski et al., 2011); to implement non-wear algorithms that might better discriminate between true non-wear and sedentary time (Choi et al., 2012); and to apply activity intensity cut-points specific to certain populations, such as children and older adults (Copeland and Esliger, 2009).

As the level of complexity in data processing methods increases, the learning curve for researchers interested in working with accelerometer data also increases. More generally, researchers with data from studies other than NHANES are tasked with writing their own software to convert the time-series data to a form suitable for statistical analysis. Such efforts are time-consuming and probably inefficient, as many different researchers are likely to write their own code for very similar purposes. The size of accelerometer files causes additional complexity. For example, the accelerometer files in NHANES 2003–2006 are 3.87 GB and contain over 1.25 billion data points.

The purpose of this paper is to describe recently developed software for processing accelerometer data. The software takes the form of two R packages: **accelerometry** for general functions (Van Domelen, 2014), and **nhanesaccel** for processing NHANES 2003–2006 data (Van Domelen et al., 2014).

Table 1: Design goals for R package **accelerometry**.

Goal	Description
1	Allow flexibility in data processing (non-wear algorithm, bouts, etc.).
2	Able to process data from uniaxial or triaxial accelerometers worn at the hip, wrist, or other position.
3	Run efficiently so that large datasets can be processed quickly.
4	Generate a wide variety of physical activity variables, including those that can be used to study weekday/weekend and time-of-day effects.
5	Use reasonable defaults for function inputs so that users with minimal knowledge of accelerometry can still process their data adequately.
6	Free.

R package ‘accelerometry’

Overview

The package **accelerometry** has a collection of functions for performing various tasks in accelerometer data processing, as well as composite functions for converting minute-to-minute uniaxial or triaxial accelerometer data into meaningful PA variables. This package is intended for researchers who wish to process accelerometer data from studies other than NHANES.

Design goals

The package was developed according to the design goals in Table 1. Computing speed was considered particularly important because users may have data on thousands or tens of thousands of participants. Such a large volume of data can take hours to process using inefficient software.

Implementation

Using R with embedded C++ code was a natural choice to meet goals 3 and 6, and the functions in the package were written to achieve goals 1–2 and 4–5. The package contains eleven functions for performing various steps in accelerometer data processing. These functions are summarized in Table 2. For seven of the functions, C++ code was added via the **Rcpp** package (Eddelbuettel and François, 2011; Eddelbuettel, 2013) to improve efficiency.

Most users will only interact with `accel.process.uni` or `accel.process.tri`. These functions internally call the other functions, and have inputs to control the non-wear and activity bout algorithms, count cut-points for activity intensities, and other options. Notably, `accel.process.tri` allows the user to choose which accelerometer axis to use for non-wear detection, activity bouts, and intensity classification. This is important because vertical-axis counts are almost always used for intensity classification, but triaxial data is more sensitive to small movements and therefore well-suited for non-wear detection (Choi et al., 2012).

Version 2.2.4 of **accelerometry** is currently available on CRAN. The package requires R version 3.0.0 or above due to its dependency on **Rcpp**, which itself requires R 3.0.0 or above. A data dictionary for the variables generated by the functions in **accelerometry** is available online (Van Domelen, 2013).

Examples

Suppose a researcher wishes to process triaxial data collected from an ActiGraph GT3X+ device (ActiGraph, Pensacola, FL) worn at the hip for a particular study participant. The first step is to load a data file with the accelerometer count values in one-minute intervals (“epochs”) into R. After transferring data from the accelerometer to a computer via USB cable, the researcher can use ActiGraph’s ActiLife software (ActiGraph, 2014) to create a .csv file and load it into R using `read.csv`, or use the function `gt3xAccFile` in the R package **pawacc** (Geraci, 2014; Geraci et al., 2012) to load the accelerometer file directly into R. Notably, data collected in shorter than one-minute epochs can still be processed using **accelerometry**, it just has to be converted to one-minute epochs first. This can

Table 2: Brief description of functions in **accelerometry**, and whether each function uses C++ code.

Function	Description	C++
<code>accel.artifacts</code>	Replace extreme counts with mean of neighboring values.	Yes
<code>accel.bouts</code>	Identify bouts of physical activity. Can specify minimum count value, minimum length, and tolerance.	Yes
<code>accel.intensities</code>	Compute number of minutes with counts in various user-defined intensity levels and counts accumulated from each intensity level.	No
<code>accel.sedbreaks</code>	Count number of sedentary breaks.	Yes
<code>accel.weartime</code>	Identify periods when participant is not wearing the accelerometer.	Yes
<code>blockaves</code>	Calculate average of non-overlapping segments of data.	Yes
<code>movingaves</code>	Calculate moving averages.	Yes
<code>rle2</code>	Run length encoding. Often much faster than base function <code>rle</code> , and records value, length, and indices for each run.	Yes
<code>inverse.rle2</code>	Convert matrix created by <code>rle2</code> back to vector form.	No
<code>accel.process.uni</code>	Composite function to process uniaxial accelerometer data.	No
<code>accel.process.tri</code>	Composite function to process triaxial accelerometer data.	No

be done in ActiLife or in R using one of several available functions (e.g. `blockaves` in **accelerometry**, `aggAccFile` in **pwacc**, or `dataCollapser` in **PhysicalActivity**; Choi et al. 2011a).

Once the data is in R, the user should have a four-column matrix or data frame where the columns represent counts in the vertical axis, anteroposterior (AP) axis, and mediolateral (ML) axis, as well as steps. A sample matrix with this format is included in the **accelerometry** package. The next step is to install **accelerometry** and call `accel.process.tri`. The following code installs the package and processes the sample dataset in two ways: first using default settings, and then using a different algorithm for non-wear detection, and requesting a larger set of PA variables.

```
# Install and load accelerometry package
install.packages("accelerometry")
library("accelerometry")

# Load four-column matrix with counts and steps over 7 days
data(tridata)

# Generate basic PA variables using default settings
dailyPA1 <- accel.process.tri(counts.tri = tridata[, 1:3], steps = tridata[, 4])

# Request full set of PA variables, and use triaxial vector magnitude for non-wear
# detection rather than vertical axis, with 90-minute rather than 60-minute window
dailyPA2 <- accel.process.tri(counts.tri = tridata[, 1:3], steps = tridata[, 4],
                             brevity = 3, nonwear.axis = "mag", nonwear.window = 90)

# Check variable names for dailyPA1 and dailyPA2
colnames(dailyPA1)
colnames(dailyPA2)

# Print contents of dailyPA1 and first 15 variables in dailyPA2
dailyPA1
dailyPA2[, 1:15]

# Calculate average for cpm_vert from dailyPA1 and dailyPA2
mean(dailyPA1[, "cpm_vert"])
mean(dailyPA2[, "cpm_vert"])
```


Comparing the two datasets, we see that `dailyPA1` has 15 variables for each of the seven days of monitoring, while `dailyPA2` has 124. The `brevity` input was not specified in the first function call, so the default was used (`brevity = 1`). This resulted in only basic variables being generated: participant ID (`id`), day of week (`day`), whether the day was deemed valid for analysis (`valid_day`), wear time minutes (`valid_min`), total counts in each accelerometer axis (`counts_vert`, ..., `counts_mag`), average counts during wear time for each accelerometer axis (`cpm_vert`, ..., `cpm_mag`), and number of steps (`steps`). In contrast, setting `brevity = 3` for `dailyPA2` resulted in an additional 109 variables being generated. These extra variables quantify various aspects of daily activity that may be of interest to more experienced PA researchers. For example, one may want to test whether prolonged periods of sedentariness (`sed_bouted_60min`) are associated with health outcomes independent of total PA (`cpm_vert`), or compare daily patterns of PA across various demographics (`cpm_hour1`, ..., `cpm_hour24`). A data dictionary for these variables is available online ([Van Domelen, 2013](#)).

Looking at `dailyPA1`, it is interesting that `counts_vert` and `counts_ap` were very similar on all days except day 6, when `counts_ap` was more than three times greater than `counts_vert`. It seems likely that the participant engaged in some non-walking activity on this day that involved more anteroposterior (i.e. front-back) motion than vertical motion. This is supported by the participant having an unusually low number of steps on this day, but only slightly lower than average `counts_ap`.

A 60-minute non-wear algorithm based on only vertical-axis counts was used to create `dailyPA1`, while a 90-minute algorithm based on triaxial counts was used to create `dailyPA2`. The first algorithm identified some non-wear time on days 5 and 6, while the second non-wear algorithm did not mark any time on any day as non-wear. Some disagreement between different non-wear algorithms is to be expected, and different researchers may have different preferences based on validation studies or their own experiences processing accelerometer data. Regardless, it makes little difference which algorithm is used for this particular participant. Average counts per minute during wear time (`cpm_vert`) for the seven days of monitoring, a standard measure of total PA, was 142.7 using the first non-wear algorithm and 142.0 using the second.

Adding `return.form = 1` to the function calls to `accel.process.tri` would result in daily averages being returned rather than separate values for each day of monitoring. This format is usually more useful for statistical analysis.

For further information on **accelerometry**, the package documentation includes a description of all of the functions and their inputs and has some additional examples ([Van Domelen, 2014](#)). The code from this section is also available in the package as a demo that can be executed by running `demo("acceljournal")`.

R package ‘nhanesaccel’

Overview

The package **nhanesaccel** is intended for researchers who want to use the objectively measured PA data in NHANES 2003–2006. Its primary function, `nhanes.accel.process`, converts minute-to-minute accelerometer data from NHANES participants into useful PA variables for statistical analysis.

Design goals

Design goals for **nhanesaccel** are shown in Table 3. Speed and flexibility were primary considerations for this package.

Implementation

The main function in **nhanesaccel** is `nhanes.accel.process`. This function relies heavily on the functions in **accelerometry**, and has similar inputs as `accel.process.uni` for controlling data processing methods. There are 31 function inputs controlling methods for non-wear detection, activity bouts, inclusion criteria, and the number of PA variables generated. These are described in the package documentation files ([Van Domelen et al., 2014](#)). A data dictionary for the variables generated by `nhanes.accel.process` is available online ([Van Domelen, 2013](#)).

The use of C++ code in the functions in **accelerometry** made it possible for `nhanes.accel.process` in **nhanesaccel** to process the full dataset of 14,631 participants in NHANES 2003–2006 in less than one minute. By including NHANES data in the package, there is no need for the user to download the large raw data files from the NHANES website. Adjusted sample weights, which are needed to ensure that the subset of participants with usable accelerometer data are weighted to match the demographics of the United States as a whole, are calculated by the function `nhanes.accel.reweight`. This function

Table 3: Design goals for R package **nhanesaccel**.

Goal	Description
1	Process NHANES 2003–2006 accelerometer data in less than 10 minutes.
2	Generate file for statistical analysis without directly modifying any code.
3	Produce .csv file that can be imported into any software package for analysis.
4	Do not require user to download 3.87 GB data files to the personal computer.
5	Calculate adjusted NHANES sample weights based on subset of participants with usable data, which are needed for appropriate statistical analysis.
6	Free.
7	Option to replicate methods used in the NCI's SAS programs, allowing researchers to use NCI's methods while taking advantage of goals 1–4 and 6.

operates very similarly to the NCI's SAS program `reweight.pam` (National Cancer Institute, 2007), and is called internally by `nhanes.accel.process`.

Version 2.1.1 of **nhanesaccel** is currently available on R-Forge. Due to its size (88.1 MB) it cannot be hosted on CRAN. The package requires R version 3.0.0 or above.

Examples

Users can install **nhanesaccel** via the `install.packages` function in R. Mac and Linux users and Windows users not running the most recent version of R (3.1 as of November 28, 2014) need to set `type = "source"` to install from source files rather than binaries.

```
# Install accelerometry package (if not already installed)
install.packages("accelerometry")

# Install nhanesaccel on Windows running most recent version of R
install.packages("nhanesaccel", repos = "http://R-Forge.R-project.org")

# Install nhanesaccel on Mac or Linux or on Windows running earlier version of R
install.packages("nhanesaccel", repos = "http://R-Forge.R-project.org",
                type = "source")
```

Once the package is installed, the user can process the NHANES 2003–2006 dataset by loading the package and then calling `nhanes.accel.process`.

```
# Load nhanesaccel package
library("nhanesaccel")

# Process NHANES 2003–2006 data using default settings
nhanes1 <- nhanes.accel.process()

# Examine summary data for first 5 participants
nhanes1[1:5, ]
```

After a short period of time, the function returns a data frame named `nhanes1` with daily averages for basic PA variables. Alternatively, the user may want to write a .csv file that can be imported into a different statistical software package for analysis. This can be achieved by adding the input `write.csv = TRUE` to the `nhanes.accel.process` function call. Unless otherwise specified via the `directory` input, the .csv file would be written to the current working directory, which can be seen by running `getwd()` and set by running `setwd("<location>")`.

At this point the user could close R and switch to his or her preferred software package for statistical analysis. For example, the user could load the .csv file into SAS, merge in the Demographics files (which can be downloaded from the NHANES website), and use survey procedures in SAS to test for associations between demographic variables and PA.

Here we briefly illustrate typical data processing and analysis in R. Suppose we wish to test the hypothesis that American youth are more active on weekdays than on weekend days. First, we process

the accelerometer data from NHANES 2003–2006 using three non-default inputs. The first two inputs require that participants have at least one valid weekday and one valid weekend day (rather than just one valid day overall), and the third requests that PA averages are calculated for weekdays and weekend days separately.

```
# Process NHANES 2003–2006 data, requiring at least one valid weekday and weekend day
nhanes2 <- nhanes.accel.process(valid.week.days = 1, valid.weekend.days = 1,
                              weekday.weekend = TRUE)
```

```
# Get dimension and variable names for nhanes2
dim(nhanes2)
names(nhanes2)
```

The data frame `nhanes2` has 14,631 rows and 17 columns. Each row summarizes the PA of one NHANES participant. The first few variables are participant ID (`seqn`), NHANES year (`wave`; 1 for 2003–2004, 2 for 2005–2006), number of valid days of monitoring (`valid_days`, `valid_week_days`, and `valid_weekend_days`), and whether the participant has valid data for statistical analysis (`include`). Then we have daily averages for accelerometer wear time (`valid_min`), counts (`counts`), and counts per minute of wear time (`cpm`), for all valid days and for weekdays (`wk_prefix`) and weekend days (`we_prefix`) separately.

The variable `cpm` is a standard measure of total PA. We will compare weekday `cpm` (`wk_cpm`) and weekend `cpm` (`we_cpm`) in participants age 6 to 18 years to address our hypothesis. To get age for each participant, we merge a demographics dataset (included in `nhanesaccel`) to the `nhanes2` data frame. Then we calculate for each participant the percent difference between `wk_cpm` and `we_cpm`. Positive values for `cpm_diff` indicate greater `cpm` on weekdays than on weekend days.

```
# Load demographics data and merge with nhanes2
data(dem)
nhanes2 <- merge(x = nhanes2, y = dem)
```

```
# Calculate percent difference between weekday and weekend CPM for each participant
nhanes2$cpm_diff <- (nhanes2$wk_cpm - nhanes2$we_cpm) /
  ((nhanes2$wk_cpm + nhanes2$we_cpm) / 2) * 100
```

Now we are almost ready for statistical analysis. Because NHANES uses a complex multi-stage probability sampling design, analyzing the data as a simple random sample would result in incorrect inference. We need to use functions in the `survey` package (Lumley, 2014, 2004) rather than base R functions like `t.test` and `glm`. For those who are not familiar with concepts in survey analysis, reference materials and tutorials are available (Lumley, 2004, 2012, 2010).

Here we create a survey object using design features of NHANES.

```
# Create survey object called hanes
hanes <- svydesign(id = ~ sdmvpsu, strata = ~ sdmvstra, weight = ~ wtmecl4yr_adj,
                 data = nhanes2, nest = TRUE)
```

Next we generate a figure comparing weekday and weekend PA in participants age 6 to 18 years. We use functions in the `survey` package to calculate mean (95% confidence interval) for `cpm_diff` in one-year age increments, then plot the results.

```
# Calculate mean (SE) and 95% CI's for cpm_diff for ages 6 to 18 years
mean.diff <- svyby(~ cpm_diff, by = ~ ridageyr, design = subset(hanes, ridageyr <= 18),
                  FUN = svymean, na.rm = TRUE)
ci <- confint(mean.diff)
```

```
# Plot means and CI's for cpm_diff by age
plot(x = 6:18, y = mean.diff[, 2], main = "CPM on Weekdays vs. Weekends",
     ylim = c(-30, 30), pch = 19, ylab = "Perc. diff. (mean +/- 95% CI)",
     xlab = "Age (years)", cex = 0.8, cex.axis = 0.85, cex.main = 1.5,
     cex.lab = 1.1, xaxt = "n")
axis(side = 1, at = 6:18, cex.axis = 0.85)
segments(x0 = 6:18, y0 = ci[, 1], x1 = 6:18, y1 = ci[, 2], lwd = 1.3)
abline(h = 0, lty = 2)
```

Figure 1 shows that PA is similar on weekdays and weekends from age 6 to 12 years, but 10–20% greater on weekdays from age 13 to 18 years (all $p < 0.05$). These results support our hypothesis that

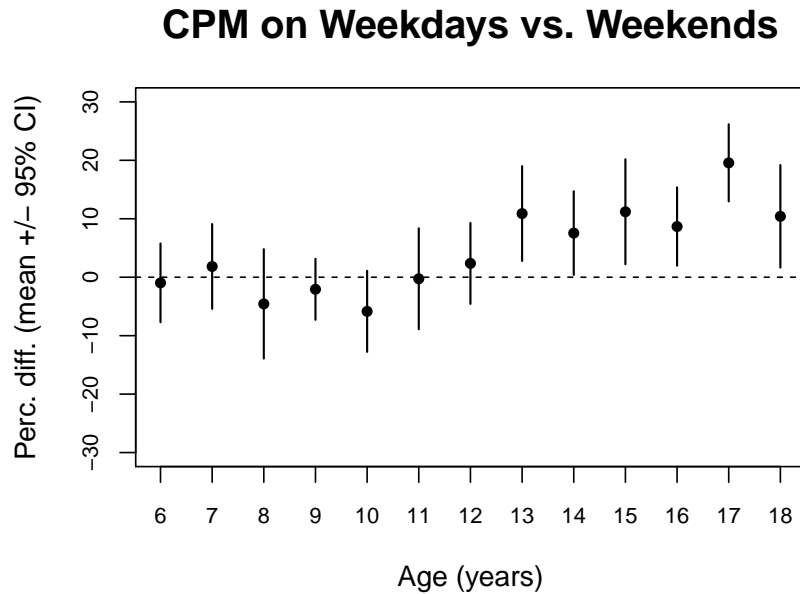


Figure 1: Mean (95% confidence interval) for percent difference between weekday and weekend CPM for NHANES 2003–2006 participants ages 6 to 18 years. Positive values indicate greater physical activity on weekdays.

American youth are more active on weekdays than on weekend days, but it appears that this is only the case for teenagers. Further analyses could try to identify the reason for the weekday/weekend difference in adolescents, for example by incorporating data on sports participation and time spent watching television for each age group.

Going back to the data processing step, users may wish to implement some non-default settings. To illustrate, the following code generates the full set of PA variables (203 variables), requires at least four valid days of monitoring, and uses a 90-minute rather than 60-minute minimum window for non-wear time.

```
# Process NHANES 2003-2006 data with four non-default inputs
nhanes3 <- nhanes.accel.process(brevity = 3, valid.days = 4, nonwear.window = 90,
                               weekday.weekend = TRUE)
```

The next example provides code to replicate the methods used in the NCI's SAS programs to process NHANES 2003–2006 data. In the first function call, we explicitly set each function input as needed to replicate the NCI's methods; in the second, we use the `nci.methods` argument as a shortcut.

```
# Specify count cut-points for moderate and vigorous intensity in youth
youthmod <- c(1400, 1515, 1638, 1770, 1910, 2059, 2220, 2393, 2580, 2781, 3000, 3239)
youthvig <- c(3758, 3947, 4147, 4360, 4588, 4832, 5094, 5375, 5679, 6007, 6363, 6751)
```

```
# Process NHANES 2003-2006 data using NCI's methods
nci1 <- nhanes.accel.process(waves = 3, brevity = 2, valid.days = 4,
                            youth.mod.cuts = youthmod, youth.vig.cuts = youthvig,
                            cpm.nci = TRUE, days.distinct = TRUE, nonwear.tol = 2,
                            nonwear.tol.upper = 100, nonwear.nci = TRUE,
                            wear.time.maximum = 1440, active.bout.tol = 2,
                            active.bout.nci = TRUE, artifact.thresh = 32767,
                            artifact.action = 3)
```

```
# Repeat, but use nci.methods input for convenience
nci2 <- nhanes.accel.process(waves = 3, brevity = 2, nci.methods = TRUE)
```

```
# Verify that nci1 and nci2 are equivalent
all(nci1 == nci2, na.rm = TRUE)
```

Table 4: Mean (standard deviation) for time to process NHANES 2003–2004 data using **nhanesaccel** and using the NCI’s SAS programs, based on five trials for each method.

nhanesaccel (defaults)	nhanesaccel (NCI methods)	NCI’s SAS programs
8.64 (0.02) s	17.34 (0.04) s	25.03 (0.79) min

While we prefer using the default function inputs, there are some compelling reasons to use the NCI’s methods instead. Doing so is simple, easy to justify in manuscripts, and ensures consistency with prior studies that have used the NCI’s SAS programs. On the other hand, certain aspects of the NCI’s methods are suboptimal. For example, the non-wear algorithm is applied to each day of monitoring separately, which results in misclassification of non-wear time as sedentary time when participants remove the accelerometer between 11 pm and midnight to go to sleep (Winkler et al., 2012). Of course, it is up to the user to decide whether to use the default settings, the NCI’s methods, or some other combination of function inputs. Certainly many different approaches can be justified.

The code from this section of the paper is included in **nhanesaccel** as a demo that can be executed by running `demo("hanesjournal")`.

Processing times vs. NCI’s SAS programs

One of the challenges of working with the NHANES accelerometer dataset is its size. A program written in R or MATLAB using standard loops to generate variables for nearly 15,000 participants can take hours to process. Efficiency was therefore a high priority in the development of **nhanesaccel**.

Processing times for **nhanesaccel** and the NCI’s SAS programs were compared using R version 3.1.0 and SAS Version 9.4, respectively, on a Dell OptiPlex 990 desktop computer with Intel Core i7-2600 CPU at 3.4 GHz and 8 GB of RAM. The `system.time` function in R was used to record processing times for **nhanesaccel**, and “real time” output from the SAS log file was used for the NCI’s programs. In both cases, elapsed times rather than CPU times were recorded. Results are shown in Table 4.

The package **nhanesaccel** was 174 times faster than the NCI’s SAS programs under default settings, and 87 times faster using settings that replicate the NCI’s methods. For **nhanesaccel**, the “NCI methods” settings have longer processing times than the defaults because the NCI’s algorithms for non-wear and activity bout detection are more computationally intensive than the defaults.

Equivalence between **nhanesaccel** (NCI methods) and NCI’s SAS programs was confirmed by direct comparison of the files generated. The only difference was that the NCI’s SAS programs had data on nine participants with potentially unreliable data, whereas **nhanesaccel** excluded those participants entirely. Excluding this data is considered acceptable by the NCI (National Cancer Institute, 2007).

Both the NCI’s SAS programs and **nhanesaccel** require some one-time steps prior to processing. The SAS programs require downloading and unzipping the data files, converting from .xpt format to SAS datasets, and downloading four SAS scripts and making minor changes to each. This process takes about ten minutes. As for **nhanesaccel**, installation can take a minute or two due to its large size.

Processing the full NHANES 2003–2006 using **nhanesaccel** takes approximately twice the time to process just NHANES 2003–2004. In five trials, mean (standard deviation) processing times were 17.57 (0.01) s using defaults and 40.26 (0.06) s using the NCI’s methods.

Discussion

The **nhanesaccel** package should increase accessibility to the objectively measured PA data in NHANES 2003–2006. The software is free, efficient, and allows flexibility in data processing without writing original code to process over a billion data points. Proficiency in R is not required, as researchers can obtain a file for statistical analysis in SAS or another software package with just three lines of R code (install package, load package, call `nhanes.accel.process`). For researchers with data from studies other than NHANES, the functions in **accelerometry** should greatly simplify the process of converting time-series count data into meaningful PA variables.

There are several limitations of the packages presented in this article. First, the functions are not immediately useful for processing accelerometer data collected in less than one-minute epochs. Researchers with 1-s, 30-Hz, or 80-Hz data could convert the data to one-minute epochs and then use the functions in **accelerometry**, but doing so would sacrifice any additional information contained in the higher resolution data. Also, the default settings for the functions in **accelerometry** and **nhanesaccel** were chosen for data collected from accelerometers worn at the hip. Users who wish

to process wrist data could still use the functions, but would have to take care to adjust function parameters, particularly `int.cuts`.

Another limitation is that the NHANES 2003–2006 dataset is 8–11 years old, and the data has been available for about six years. However, NHANES 2003–2006 remains the most recent study with objectively measured PA on a nationally representative sample of Americans. Although population activity levels may have changed since 2003–2006, associations between PA and outcomes should be relatively constant over time. Additionally, data on deaths will continue to be updated, which will enable future studies on PA and 5- or 10-year mortality. There are likely still many opportunities for useful contributions to the PA literature from this dataset.

There are several other R packages for processing accelerometer data. The **PhysicalActivity** package (Choi et al., 2011b) has functions for converting accelerometer data to longer epochs (`dataCollapser`), plotting accelerometer data (`plotData`), and implementing a non-wear algorithm (`wearingMarking`). Briefly, this algorithm uses a 90- rather than 60-minute window, and allows one or two non-zero count values provided that they are surrounded by 30 minutes of zero counts on both sides. Variants of this non-wear algorithm can be implemented by adjusting function inputs. The **GGIR** package (van Hees et al., 2014) has functions for processing high-frequency triaxial data from certain model accelerometers, and has a function for generating summary statistics. The variables that **GGIR** generates are mostly indicators of overall PA, such as mean acceleration over the full day or during time periods within the day. Finally, **pawacc** has functions for converting to longer epochs, detecting non-wear and activity bouts, generating summary statistics, and plotting data. The **pawacc** package can also be used to read accelerometer files into R without using ActiGraph's ActiLife software. To our knowledge, **nhanesaccel** is the only shared software currently available for processing the data from NHANES 2003–2006 other than the NCI's SAS programs.

Looking forward, there is a great need for software to process high-frequency accelerometer data. In particular, wrist-worn accelerometers are being used in NHANES 2011–2014, with the devices recording triaxial data at 80-Hz (Troiano, 2012). Analyzing this data will be challenging, as there will be over 145 million data points per participant, or about two trillion data points total. Aside from the sheer volume of data, methods for generating meaningful activity variables from high-frequency triaxial data are still being developed. It will be interesting to see how this data will be made available to the public, and whether the NCI will provide data processing software similar to the SAS programs for NHANES 2003–2006. Provided that the raw data is made available in some form, we intend to add functions to **nhanesaccel** or develop a separate R package to process NHANES 2011–2014 data.

We hope that **accelerometry** and **nhanesaccel** will make it easier for researchers to work with accelerometer data, and particularly the NHANES 2003–2006 dataset. Researchers who have any problems using either package, or have suggestions for additional features, should not hesitate to contact us.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0940903.

Bibliography

- ActiGraph. ActiLife 6 Data Analysis Software. Actigraph, Pensacola, Florida, United States. URL <http://www.actigraphcorp.com/product-category/software>, 2014. [p53]
- A. Bankoski, T. B. Harris, J. J. McClain, R. J. Brychta, P. Caserotti, K. Y. Chen, D. Berrigan, R. P. Troiano, and A. Koster. Sedentary activity associated with metabolic syndrome independent of physical activity. *Diabetes Care*, 34(2):497–503, 2011. [p52]
- V. Carson and I. Janssen. Volume, patterns, and types of sedentary behavior and cardio-metabolic health in children and adolescents: A cross-sectional study. *BMC Public Health*, 11(274), 2011. [p52]
- L. Choi, Z. Liu, C. E. Matthews, and M. S. Buchowski. *PhysicalActivity: Process Physical Activity Accelerometer Data*, 2011a. URL <http://CRAN.R-project.org/package=PhysicalActivity>. R package version 0.1-1. [p54]
- L. Choi, Z. Liu, C. E. Matthews, and M. S. Buchowski. Validation of accelerometer wear and nonwear time classification algorithm. *Medicine & Science in Sports & Exercise*, 43(2):357–364, 2011b. [p60]

- L. Choi, S. C. Ward, J. F. Schnelle, and M. S. Buchowski. Assessment of wear/nonwear time classification algorithms for triaxial accelerometer. *Medicine & Science in Sports & Exercise*, 44(10):2009–2016, 2012. [p52, 53]
- R. C. Colley, D. Garriguet, I. Janssen, C. L. Craig, J. Clarke, and M. S. Tremblay. Physical activity of Canadian children and youth: Accelerometer results from the 2007 to 2009 Canadian Health Measures Survey. *Health Reports*, 22(1):1–9, 2011. [p52]
- J. L. Copeland and D. W. Eslinger. Accelerometer assessment of physical activity in active, healthy older adults. *Journal of Aging and Physical Activity*, 17(1):17–30, 2009. [p52]
- D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7. [p53]
- D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL <http://www.jstatsoft.org/v40/i08/>. [p53]
- M. Geraci. *pawacc: Physical Activity with Accelerometers*, 2014. URL <http://CRAN.R-project.org/package=pawacc>. R package version 1.2. [p53]
- M. Geraci, C. Rich, F. Sera, M. Cortina-Borja, L. J. Griffiths, and C. Dezateux. Technical report on accelerometry data processing in the Millennium Cohort Study. London, UK: University College London, 2012. URL <http://discovery.ucl.ac.uk/1361699>. [p53]
- M. Hagstromer, P. Oja, and M. Sjostrom. Physical activity and inactivity in an adult population assessed by accelerometry. *Medicine & Science in Sports & Exercise*, 39(9):1502–1508, 2007. [p52]
- G. N. Healy, C. E. Matthews, D. W. Dunstan, E. A. Winkler, and N. Owen. Sedentary time and cardio-metabolic biomarkers in U.S. adults: NHANES 2003–2006. *European Heart Journal*, 32(5): 590–597, 2011. [p52]
- T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. URL <http://www.jstatsoft.org/v09/i08/>. [p57]
- T. Lumley. *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons, Inc., Hoboken, NJ, 2010. [p57]
- T. Lumley. Survey analysis in R. Website for ‘survey’ package. URL <http://R-survey.R-Forge.R-project.org/survey>, 2012. [p57]
- T. Lumley. *survey: Analysis of Complex Survey Samples*, 2014. URL <http://CRAN.R-project.org/package=survey>. R package version 3.30. [p57]
- National Cancer Institute. Risk factor monitoring and methods: SAS programs for analyzing NHANES 2003–2004 accelerometer data. URL http://riskfactor.cancer.gov/tools/nhanes_pam, 2007. [p52, 56, 59]
- National Instruments. *LabVIEW: Laboratory Virtual Instrument Engineering Workbench*. Austin, Texas, United States, 2014. URL <http://www.ni.com/labview>. [p52]
- S. B. Sisson, S. M. Camhi, T. S. Church, C. Tudor-Locke, W. D. Johnson, and P. T. Katzmarzyk. Accelerometer-determined steps/day and metabolic syndrome. *American Journal of Preventive Medicine*, 38(6):575–582, 2010. [p52]
- The MathWorks, Inc. *MATLAB – The Language of Technical Computing, Version R2014b*. The MathWorks, Inc., Natick, Massachusetts, 2014. URL <http://www.mathworks.com/products/matlab/>. [p52]
- R. P. Troiano. Physical activity among children and adolescents: Data from National Health and Nutrition Examination Survey (NHANES) 2003–2006. URL http://www.cdc.gov/nchs/ppt/nchs2012/SS-15_TROIANO.pdf, 2012. [p60]
- R. P. Troiano, D. Berrigan, K. W. Dodd, L. C. Mâsse, T. Tilert, M. McDowell, et al. Physical activity in the United States measured by accelerometer. *Medicine & Science in Sports & Exercise*, 40(1):181, 2008. [p52]
- C. Tudor-Locke, M. M. Brashear, W. D. Johnson, and P. T. Katzmarzyk. Accelerometer profiles of physical activity and inactivity in normal weight, overweight, and obese U.S. men and women. *The International Journal of Behavioral Nutrition and Physical Activity*, 7(60), 2010. [p52]

- C. Tudor-Locke, S. M. Camhi, and R. P. Troiano. A catalog of rules, variables, and definitions applied to accelerometer data in the National Health and Nutrition Examination Survey, 2003-2006. *Preventing Chronic Disease*, 9:E113, 2011. [p52]
- D. R. Van Domelen. Personal/professional website. URL <http://sites.google.com/site/danevandomelen>, 2013. [p53, 55]
- D. R. Van Domelen. *accelerometry: Functions for Processing Uniaxial Minute-to-Minute Accelerometer Data*, 2014. URL <http://CRAN.R-project.org/package=accelerometry>. R package version 2.2.4. [p52, 55]
- D. R. Van Domelen, W. S. Pittard, and T. B. Harris. *nhanesaccel: Process Accelerometer Data from NHANES 2003–2006*, 2014. URL <http://R-Forge.R-project.org/projects/nhanesaccel/>. R package version 2.1.1. [p52, 55]
- V. T. van Hees, Z. Fang, and J. H. Zhao. *GGIR: A Package to Process Multi-Day Raw Accelerometer Data*, 2014. URL <http://CRAN.R-project.org/package=GGIR>. R package version 1.0-1. [p60]
- E. A. H. Winkler, P. A. Gardiner, B. K. Clark, C. E. Matthews, N. Owen, and G. N. Healy. Identifying sedentary time using automated estimates of accelerometer wear time. *British Journal of Sports Medicine*, 46:436–442, 2012. [p59]

Dane R. Van Domelen
Department of Biostatistics and Bioinformatics
Rollins School of Public Health
Emory University
1518 Clifton Road, Room 323
Atlanta, GA 30322
United States
dvandom@emory.edu

W. Stephen Pittard
Department of Biostatistics and Bioinformatics
Rollins School of Public Health
Emory University
1518 Clifton Road, Room 366
Atlanta, GA 3022
United States
wsp@emory.edu

Applying *spartan* to Understand Parameter Uncertainty in Simulations

by Kieran Alden, Mark Read, Paul S Andrews, Jon Timmis and Mark Coles

Abstract In attempts to further understand the dynamics of complex systems, the application of computer simulation is becoming increasingly prevalent. Whereas a great deal of focus has been placed in the development of software tools that aid researchers develop simulations, similar focus has not been applied in the creation of tools that perform a rigorous statistical analysis of results generated through simulation: vital in understanding how these results offer an insight into the captured system. This encouraged us to develop *spartan*, a package of statistical techniques designed to assist researchers in understanding the relationship between their simulation and the real system. Previously we have described each technique within *spartan* in detail, with an accompanying immunology case study examining the development of lymphoid tissue. Here we provide a practical introduction to the package, demonstrating how each technique is run in R, to assist researchers in integrating this package alongside their chosen simulation platform.

Introduction

Utilising mathematical or agent-based computational models to further understand the dynamics of complex systems is becoming increasingly popular. With the benefits of performing simulated experiments where the conditions are fully controlled, or where ethical and financial constraints are avoided, researchers can generate a predictive tool that has the capacity to inform future studies of the captured system. To aid the adoption of this approach, much focus has been placed on providing researchers with software tools through which they can generate a simulation (Ivanyi et al., 2007; Wilensky, 1999; Luke, 2005; Meier-Schellersheim et al., 2006). Although these simulation development platforms provide easy to use functionality for generating a large set of simulation results, these platforms do not equip the researcher with the means of statistically analysing these results. However, undertaking a full statistical analysis to gain a full appreciation of the way the simulated biological system behaves is vital.

Our recent work utilised agent-based modelling to further understand the development of secondary lymphoid organs in the gastrointestinal tract. Thorough statistical analyses of simulation behaviour demonstrated that the simulator produced emergent cell behaviour that is statistically similar to that observed in the laboratory (Patel et al., 2012; Alden et al., 2012). As the analyses suggested this is the case, we were able to utilise our tool to perform *in silico* experimentation to inform future laboratory based studies. In order to do this, fully understanding the relationship between the simulation and the real-world system that has been captured is vital: can a result attributed to the real-world system, or is it an artefact of simulation implementation or uncertainty in parameter values? Such understanding is key where the aim is to transfer an hypothesis generated through simulation to the real-world. This led us to develop *spartan* (Simulation Parameter Analysis R Toolkit Application) (Alden et al., 2013b), a package of statistical techniques that aid the researcher in understanding this relationship such that a simulation can provide novel insight into the system being studied. We do note that in addition to *spartan*, several other packages for sensitivity analysis are also available (Lamboni et al., 2013; Dancik, 2013; Pujol et al., 2014). However *spartan* has been compiled to equip developers of biological simulations with the statistical techniques necessary to maximise the potential of the simulation platform they generate. This package can be applied to agent-based simulations such as ours, and to traditional ordinary or partial differential equation models. To encourage adoption of these techniques by the community, *spartan* is supported by detailed tutorials and a growing number of case studies: the first of which is demonstrated in this paper. Although the development of *spartan* has resulted from a number of such biological modelling studies, we do not rule out the potential application of the package outside of the biological sciences domain.

spartan has been implemented in R, released under a GPLv2 license, and is available from CRAN. The package is a compilation of four previously described statistical techniques (Read et al., 2012; Marino et al., 2008; Saltelli and Bollardo, 1998) that each provide an alternative means of analysing simulation data. The first technique helps a researcher ensure that a response generated by a non-deterministic simulation is a true representation of the parameter set upon which it is being run. The second is a local sensitivity analysis technique that suggests how robust simulation behaviour is to parameter perturbation, with techniques 3 and 4 being global sensitivity analysis techniques that help understand the key pathways affecting simulation behaviour. Our complementary *spartan* paper (Alden et al., 2013b) describes each technique in detail, and utilises our case study of lymphoid tissue development to demonstrate to the researcher the impact that such statistical techniques could

have in understanding their simulation behaviour. Here we take this description a stage further, and demonstrate the *application* of each technique in R. Alongside full examples of the R code and commands required to run each technique, we have provided the reader with example simulation data, available from our laboratory website (www.ycil.org.uk). As such, our description below is fully reproducible, and encourages simulation developers to adopt **spartan** alongside their chosen simulation development tool. For space reasons, we focus on the application of rather than the full implementation detail of each technique, and as such this paper should be seen as an accompaniment to our previous **spartan** publication.

Prerequisites

The following are required to run **spartan** as we describe in this paper:

- The R statistical environment, version 2.13.1 or later.
- The **spartan** (Alden et al., 2013a) R package, downloaded from CRAN
- The **lhs** (Carnell, 2012), **gplots** (Warnes et al., 2013), and **XML** (Lang, 2013) R packages, also available for download from CRAN.
- The example simulation results, available from our laboratory website (www.ycil.org.uk)

The case study

Our demonstration in this paper utilises data from our previously described agent-based lymphoid tissue development simulator (Patel et al., 2012; Alden et al., 2012). In this system two types of cells migrate into the developing gut and form patches of tissue at varying locations along the gut tract 72 hours later. These mature to form lymphoid organs that trigger an adaptive immune response to gut-based pathogens. Lab experimentation has revealed that cells behave in a different manner (in terms of their velocity and displacement over a 1 hour period) around an initial forming patch than elsewhere in the gut, for reasons not currently understood. To aid our understanding of tissue development, we adopted an agent-based simulation approach. In the example analyses demonstrated here, we are interested in two emergent cell behaviour responses: velocity and displacement, and how these are affected by the values assigned to simulation parameters that capture certain aspects of the biological system. These are captured at simulated hour twelve of tissue development: a time-point where we have experimental data which can be compared to simulation response. The simulator has six parameters for which there is uncertainty in parameter value, each previously described in depth (Alden et al., 2012). Here we demonstrate techniques that perturb the values of each of these parameters to reveal the role each has on the emergent cell behaviour responses.

Technique 1: Consistency analysis

Aleatory uncertainty is caused by inherent stochasticity within a non-deterministic simulation, and it is critical that the impact this has on simulation response is understood (Helton, 2008). This is especially the case for agent-based simulations such as our case study, where pseudo-random number generation can lead to the simulation producing different responses for identical parameter value sets. To mitigate this, a number of replicate simulation runs should be performed, achieving a representative result for a given set of parameters. **spartan** Technique 1, developed from a method described by Read et al. (2012) can be applied to stochastic simulation systems, to provide an indication of the number of simulation runs necessary to reduce this uncertainty, while allowing researchers to balance the computational resource required against accuracy of results. This technique is not applicable to deterministic simulations where identical responses are generated each time the simulation is run with a specified set of parameters.

For the full detail of the algorithm we direct the reader to our previously published work (Read et al., 2012; Alden et al., 2013b). As an overview, the Consistency Analysis Technique in **spartan** operates by comparing a number of distributions of simulation responses run under identical parameter values. Through altering the number of replicate simulation responses within each distribution, the number of runs required to ensure statistical consistency of response can be determined. In the example given in Read et al. (2012), 20 such distributions are used. Each contains a number of simulation responses. Distributions 2-20 are contrasted with distribution 1 using the Vargha-Delaney A-Test (Vargha and Delaney, 2000), a non-parametric effect magnitude test which provides a statistical measure of the difference between two distributions, suggesting that the results are consistent or the distribution requires a larger number of simulation runs. As would be assumed, increasing the number

of replicates decreases the variance between the distributions. **spartan** can be used to determine this number, while ensuring that an excess number of runs is avoided.

We can provide more detail with the aid of an exemplar application, utilising the example data available from our laboratory website (www.ycil.org.uk). The first step is to define the objects required for this analysis, as below. The comment above each object notes the reason for the declaration:

```
# Firstly, import the package
library(spartan)
# Directory where the example simulation results for this technique were extracted
FILEPATH <- "/home/user/AA"
# Sample sizes (number of simulation replicates in each distribution) to be analysed
SAMPLESIZES <- c(1, 5, 50, 100, 300)
# The simulation output measures to be analysed
MEASURES <- c("Velocity", "Displacement")
# Number of distributions being compared. Default: 20, as performed by Read et al
NUMSUBSETPERSAMPLESIZE <- 20
# Output file name containing the simulation responses.
RESULTFILENAME <- "trackedCells_Close.csv"
# Not used in this case. Useful where two result files exist (e.g.\ if tracking cells
# close and those further away, two output files could be used). Here, results in a
# second file are processed if the first is blank or does not exist.
ALTFILENAME <- NULL
# Notes the column in the CSV results file where the results start.
# Useful as it restricts what is read in to R, getting round potential errors where
# the first column contains a label
OUTPUTFILECOLSTART <- 10
# Last column of the output measure results
OUTPUTFILECOLEND <- 11
# Use this if simulation results are in CSV format.
# Last column of the output measure results
OUTPUTFILECOLEND <- 11
# File either A: created by method 1 of this technique, containing the median of each
# output measure of each simulation run in that subset, or B: The name of the provided
# single CSV file containing the simulation responses. So if you are using the CSV
# structured tutorial data, this will be the name of that CSV file.
MEDIANS_SUMMARY_FILE_NAME <- "AA_SimResponses.csv"
# The results of the A-Test comparisons of the twenty subsets for each sample size
# are stored within an output file. This parameter sets the name of this file.
# Note no file extension. Current versions of spartan output to CSV files
ATESTRESULTSFILENAME <- "AA_ATest_Scores.csv"
# A summary file is created containing the maximum and median
# A-Test values for each sample size. This parameter sets the name of this file.
SUMMARYFILENAME <- "AA_ATestMaxAndMedians.csv"
# The A-Test value either side of 0.5 which should be considered a 'large difference'
# between two sets of results. Use of 0.23 was taken from the Vargha-Delaney
# publication but can be adjusted here as necessary.
LARGEDIFFINDICATOR <- 0.23
# A-Test values above 0.5 (no difference) which should be considered as small,
# medium, and large differences between two result sets. Used in the graph
# summarising all sample sizes.
SMALL <- 0.56
MEDIUM <- 0.66
LARGE <- 0.73
# Name of the graph which summarises the analysis results for all sample sizes.
# Current versions of spartan output to pdf.
GRAPHOUTPUTFILE <- "AA_ATestMaxes.pdf"
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"
```

In this example, we are determining whether 1,5,50,100 and 300 replicates (stated in SAMPLESIZES)

of a simulation run are sufficient to produce a result that is representative of the parameter set upon which the simulation was run. Similar to the original description of this technique in [Read et al. \(2012\)](#), we utilise 20 distributions for each sample size (i.e. 20 distributions each containing 1 run, 20 more distributions each containing 5 runs, and so on up to 300). **spartan** (from Version 2.0) can process simulation output provided in two formats, both of which we describe next.

The first format consists of the folder structure seen in Figure 1(A), where the result of each simulation run is provided in an organised structure. This structure is comprised of three levels: the sample size, a folder for each distribution for that size, and a folder for the results for each run in that distribution. Such a structure is suitable in cases where the simulation results have been generated on a computer cluster, meaning the results need no post-processing once stored in the correct location. The second format consists of a sole CSV file that summarises the results of each run for each distribution, for each sample size. Each row of the CSV file should contain the sample size being analysed when this simulation was run, the distribution the run belongs to, and the median of each simulation response for that run. There are examples of both formats within the exemplar data that is available from our website (www.ycil.org.uk).

In our case study, each simulation result contains behavioural responses (velocity and displacement) for a number of cells. To compare the distributions using the A-Test, we need to calculate the median of each response, for each run, to produce a result set that summarises the runs in each distribution. To do this, we use the `aa_summariseReplicateRuns` function, which we describe in the next paragraph. Note however that this function should only be run when results are being provided in the first input format: the organised folder structure. The function iterates through these folders to summarise the simulation runs in a single CSV file, matching the format that can be provided by the second input format (the single CSV file).

For example, consider distribution 1 for a sample size of 50 runs. Each run contains the results for a set of cells that were tracked over the course of one simulated hour. For each run, **spartan** calculates the median of each simulation response (cell velocity and displacement in our case), storing these as a row in a CSV file that summarises cell behaviour across all the runs, named as stated in `MEDIANS_SUMMARY_FILE_NAME`. The appropriate set of medians is then generated for distributions 2-20, with these calculates repeated for all sample sizes being analysed.

```
aa_summariseReplicateRuns(FILEPATH, SAMPLESIZES, MEASURES, RESULTFILENAME,
    ALTFILENAME, OUTPUTFILECOLSTART, OUTPUTFILECOLEND, MEDIANS_SUMMARY_FILE_NAME,
    TIMEPOINTS, TIMEPOINTS SCALE)
```

Whether providing results in the organised folder structure or as a single CSV file, the researcher will now have their results in a format that can be processed by the methods contained in **spartan**. Where providing a single CSV file as input of simulation results, the name of this file should be stated in parameter `MEDIANS_SUMMARY_FILE_NAME`.

Statistical difference between the 20 distributions of each sample size is calculated by contrasting the responses of distributions 2-20 in turn with distribution 1, using the Vargha-Delaney A-Test, using the function below. To ease understanding of the results, a graph is automatically produced for each sample size (example in Figure 1(B)), plotting the A-Test result for the nineteen comparisons, and automatically saved in the directory specified by `FILEPATH`.

```
aa_getATestResults(FILEPATH, SAMPLESIZES, NUMSUBSETSPERSAMPLESIZE, MEASURES,
    MEDIANS_SUMMARY_FILE_NAME, ATESTRESULTSFILENAME, LARGEDIFFINDICATOR,
    TIMEPOINTS, TIMEPOINTS SCALE)
```

The A-Test returns the probability that a randomly selected sample from one distribution is larger than a randomly selected sample from another. Hence, a result of 0.5 means their medians are the same (even if their variance is not). Only the magnitude of the result away from 0.5 is important, the direction depends simply on which distribution has higher values. Larger differences between score and 0.5 indicate bigger differences between two distributions.

However, what is of real interest is the difference between distributions against number of replicate runs that is contained in each. These results are summarised using the following two functions:

```
aa_sampleSizeSummary(FILEPATH, SAMPLESIZES, MEASURES, ATESTRESULTSFILENAME,
    SUMMARYFILENAME, TIMEPOINTS, TIMEPOINTS SCALE)
```

```
aa_graphSampleSizeSummary(FILEPATH, MEASURES, 300, SMALL, MEDIUM, LARGE,
    SUMMARYFILENAME, GRAPHOUTPUTFILE, TIMEPOINTS, TIMEPOINTS SCALE)
```

The first creates a CSV file, in the folder specified in the `FILEPATH` variable, stating the maximum A-Test score across the distributions for each simulation output response, for each number of simulation

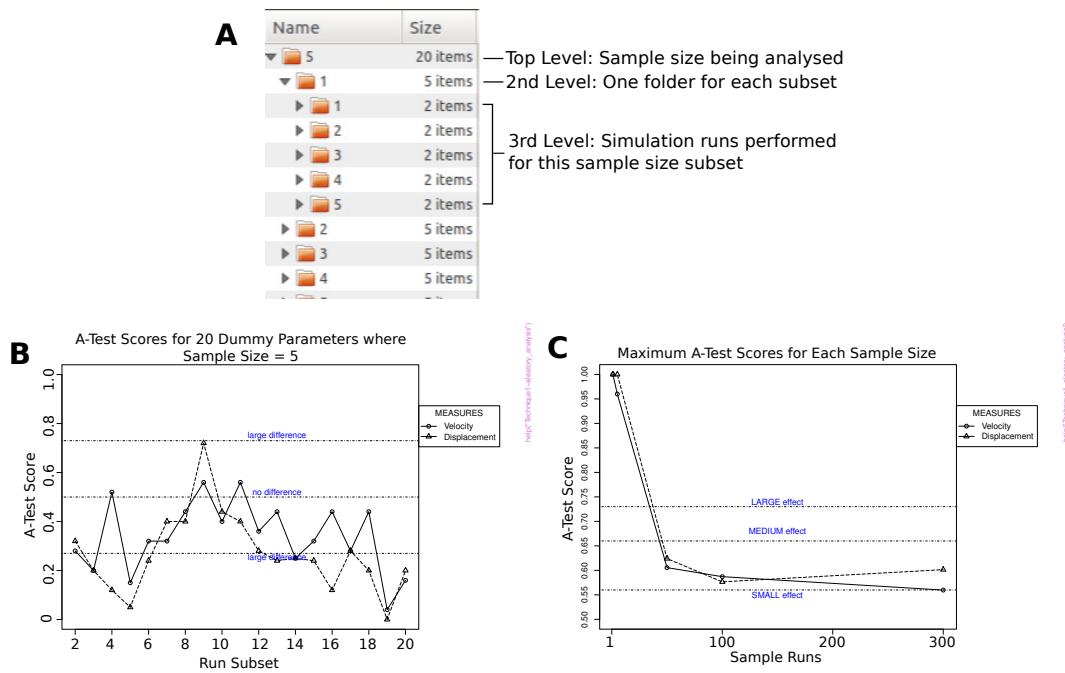


Figure 1: Use of *spartan* to mitigate aleatory uncertainty. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: A-Test scores for 20 distributions of simulation runs, with each distribution containing 5 runs. Note the substantial differences between distributions, although the same parameter values have generated these results. This suggests that 5 runs is insufficient to achieve a representative result. C: The maximum A-Test score across 20 distributions of simulation runs for sample sizes of 1,5,50,100, and 300. As direction is of no concern, the A-Test scores are normalised such that scores below 0.5 are assigned corresponding values above 0.5. B and C are reproduced from Alden et al. (2013b)

replicates examined. The second produces a plot of this result (Figure 1(C)), saved in the same folder. The objective is to find a sample size which minimises the A-Test value for all measures. As argued previously (Alden et al., 2013b), based on these results we did not deem 300 runs to be sufficiently close enough to the ‘no difference’ line for both Velocity and Displacement measures, and as such ran all our analyses with 500 runs (not included in data set due to download size). With this result taken into account, we are confident that the hypotheses we generate from simulation-based experimentation are derived from parameter values, and not inherent stochasticity caused by simulation implementation. This is an important consideration for agent-based models such as ours, and understanding the origin of changes in simulation behaviour is vital when interpreting results.

You will have noticed that each function call contains the two parameters TIMEPOINTS and TIMEPOINTSSCALE. Each function is capable of performing this analysis for multiple timepoints of a simulation, and will iterate through each as required. In addition to the steps above, such an analysis requires that:

- An array of timepoints being examined is provided in the TIMEPOINTS object, and the scale in which these are measured provided in TIMEPOINTSSCALE (see the object declarations above for an example).
- Simulation results (if provided in the folder structure input method), or the CSV files summarising simulation results, should have the timepoint at which these were produced appended to the filename. For example, in the exemplar data, there are CSV files named AA_SimResponses_12.csv, AA_SimResponses_36.csv, etc, containing cell behaviour characteristics at hours 12 and 36 respectively. The same exists in the simulation result folders, where the simulation results are named trackedCells_Close_12.csv, trackedCells_Close_36.csv, etc. It is important that the result files are named in this manner to aid *spartan* finding these result files.

Technique 2: Parameter robustness

We noted previously that for our case study, there are six parameters for which there is uncertainty in parameter value. In our case, this uncertainty applies as no biological experimental technique has

or can be used to determine a value. In other cases, such uncertainty may be derived from a lack of available data, or the generation of a hypothetical model. The Parameter Robustness technique included in **spartan** can be utilised to examine the implication this uncertainty or parameter estimation has on simulation response. If altering the value of a particular parameter from a baseline or calibrated value has a significant effect on simulator output, the simulation is highly sensitive to that parameter, and caution should be applied when both interpreting the result and establishing a value for that parameter.

Robustness analysis is performed by perturbing each parameter individually, using a 'one at a time' approach (Read et al., 2012). The value of the parameter of interest is perturbed, with all other parameters remaining at their baseline value. Simulation responses under these perturbed conditions are contrasted with responses under baseline conditions, using the Vargha-Delaney A-Test (Vargha and Delaney, 2000) described above, to determine if a scientifically significant behavioural alteration has occurred. This provides the researcher with an indication of how robust the simulation response is to parameter alteration and indicates parameter values at which simulation behaviour changes. In cases where expert knowledge is available, such parameter value windows can be contrasted to an accepted range of values, to determine if the simulation is behaving as expected. **spartan** includes methods to produce simulation parameter value sets for this statistical technique and to analyse the resultant simulation response under those conditions.

Parameter sampling

The package contains a method which can produce a set of simulation parameters for each parameter of interest. Simulations should then be run on each of the generated parameter sets. These are generated as follows:

```
# Import the package
library(spartan)
# Set a folder where the parameter value samples should be output to
FILEPATH <- "/home/user/OAT/Sampling"
# Set the names of the parameters for which values are being generated for
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold",
"chemoUpperLinearAdjust", "chemoLowerLinearAdjust",
"maxVCAMeffectProbabilityCutoff", "vcamSlope")
# The calibrated values, or baseline values, of each stated parameter
BASELINE <- c(50, 0.3, 0.2, 0.04, 0.60, 1.0)
# Parameter Value Information
# You can specify this in two ways:
# 1. The minimum and maximum of each parameter, and increment over which
# sampling should be increased.
# 2. A string list of values that parameter should be assigned in sampling
# Example of 1:
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25)
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0)
PINC <- c(10, 0.1, 0.05, 0.005, 0.05, 0.25)
PARAMVALS <- NULL
# Example of 2:
#PARAMVALS <- c("0, 50, 90", "0.10, 0.3, 0.8", "0.10, 0.25, 0.4",
"0.015, 0.04, 0.08", "0.1, 0.5, 0.9", "0.25, 1.25, 2.0, 3.0, 5.0")
# If using method 1, PARAMVALS must be set to NULL. If using method 2, PMIN,
# PMAX, and PINC must be set to NULL

oat_parameter_sampling(FILEPATH, PARAMETERS, BASELINE, PMIN, PMAX,
PINC, PARAMVALS)
```

This will produce a CSV for for each parameter, each with a file name matching the parameter which is being perturbed. In our case, 6 files are generated, and stored in the folder specified in FILEPATH. Simulations should then be run on each parameter value set in each file, and analysed using the technique described in the next section.

Analysing the results

In this section we will demonstrate how **spartan** can produce statistical information that reveals how robust the simulation behaviour is to a change in parameter value. Again we provide more detail

with the aid of an exemplar application, utilising the example data available from our laboratory website (www.ycil.org.uk). Note that to reduce the size of the download data, results for only two of the six parameters have been included. A full analysis of the six parameters has been described previously (Alden et al., 2012). The first step in this demonstration is to define the objects required for this analysis, as below. For space reasons, we have not duplicated the comments for objects that were declared in Technique 1 or in parameter sampling above: the reader should refer to these if required.

```
library(spartan)
# Folder containing the example simulation results. Make sure the folder is unzipped
FILEPATH <- "/home/user/OAT/Results"
# Array of the parameters to be analysed.
# Note only two of the six here for download size reasons
PARAMETERS <- c("chemoLowerLinearAdjust", "chemoUpperLinearAdjust")
# Similar to the sampling function discussed above, there are two ways to specify
# parameter value information in the analysis. Ensure you are using the appropriate
# method, setting these to NULL if using the alternative (see comments in sampling
# function description).
# Method 1:
PMIN <- c(0.015, 0.10)
PMAX <- c(0.08, 0.50)
PINC <- c(0.005, 0.05)
PARAMVALS<-NULL
# Method 2:
#PARAMVALS <- c("0.015, 0.02, 0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06,
#           0.065, 0.07,0.075, 0.08", "0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5")
#PMIN <- NULL; PMAX <- NULL; PINC <- NULL
BASELINE <- c(0.04, 0.2)
MEASURES <- c("Velocity", "Displacement")
# What each measure represents. Used in graphing results
MEASURE_SCALE <- c("microns/min", "microns")
RESULTFILENAME <- "trackedCells_Close.csv"
OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
ALTERNATIVEFILENAME <- NULL
# Either 1: The name of the CSV file containing all simulation output (see description
# that follows in this section) or name to give the summary file that spartan generates
CSV_FILE_NAME <- "OAT_Medians.csv"
# Number of replicate runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
# The results of the A-Test comparisons of each parameter value against that of the
# parameters baseline value are output as a file. This sets the name of this file.
# Current versions of spartan output this to a CSV file
ATESTRESULTSFILNAME <- "EgSet_ATests.csv"
# A-Test result value either side of 0.5 at which the difference between two sets of
# results is significant
ATESTSIGLEVEL <- 0.23
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"
```

In this example, we are analysing the robustness of two parameters, perturbed between 0.015-0.08 and 0.10-0.50 accordingly. Similar to Technique 1, the researcher can provide their simulation results in two formats: the organised folder structure detailed in Figure 2(A) or a single CSV file. The folder structure is comprised of three levels: the parameter being analysed, the value assigned to that parameter, and results for each run under those parameter conditions. If providing a summary of results in a single CSV file, each row should contain the value of each parameter being analysed (matching the parameters stated in PARAMETERS) and the median of each simulation response for simulation runs under those conditions. For stochastic simulations where a number of runs are required to produce a consistent result (see Technique 1), a number of results for each parameter set should be provided in the CSV file. For example, in our case study we have repeated each simulation 300 times, thus 300 lines exist in the CSV file for each parameter value set. Examples of both formats of input have been provided in the exemplar data available from our website (www.ycil.org.uk).

The first function that comprises this technique should only be run where the researcher is providing results in the organised folder structure in Figure 2(A). Similarly to Technique 1, this method iterates through the simulation results in each folder, producing the CSV file that summarises all simulation results. For our case study, this function calculates the median of each cell behaviour response for each of the 300 runs, storing this in the CSV file. Once each parameter-value pair has been processed, this CSV file is saved within the folder specified in `FILEPATH`, with the filename as stated by `CSV_FILE_NAME`. `PMIN`, `PMAX`, `PINC`, and `PARAMVALS` should be set appropriately as described in the comments of the object specifications above.

```
oat_processParamSubsets(FILEPATH, PARAMETERS, NUMRUNSPERSAMPLE, MEASURES,
  RESULTFILENAME, ALTERNATIVEFILENAME, OUTPUTCOLSTART, OUTPUTCOLEND,
  CSV_FILE_NAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS,
  TIMEPOINTS, TIMEPOINTS SCALE)
```

Whether providing results in the organised folder structure or as a single CSV file, the researcher will now have their results in a format that can be processed by the methods contained in **spartan**. Where providing a single CSV file as input of simulation results, the name of this file should be stated in parameter `CSV_FILE_NAME`.

To gain statistical information concerning the impact that a change in a single parameter has on simulation response, we call the following function.

The results of each parameter/response pair in the CSV file are processed, comparing the distribution of responses for simulations under one parameter value set conditions with responses at the calibrated/baseline value. The Vargha-Delaney A-Test described previously is utilised to perform this comparison of distributions. As an example, consider one of the parameters we are analysing in the example set, `chemoLowerLinearAdjust`. This has a baseline value of 0.04. The simulation has two output measures, velocity and displacement. The method will read all results from the CSV file where the parameter value is its calibrated value, and contrast these with results where this parameter value has been perturbed, determining the impact this value change has had on simulation response. With all parameter values examined, **spartan** produces a CSV file stating the A-Test values for each value assigned to that parameter (named as stated in object `ATESTRESULTSFILENAME`).

```
oat_csv_result_file_analysis(FILEPATH, CSV_FILE_NAME, PARAMETERS, BASELINE,
  MEASURES, ATESTRESULTFILENAME, PMIN, PMAX, PINC,
  PARAMVALS, TIMEPOINTS, TIMEPOINTS SCALE)
```

To ease assessment of these results, **spartan** can then produce a plot for each parameter, showing the difference in simulation behaviour across the parameter value space. An example of such a plot can be seen in Figure 2(B).

```
oat_graphATestsForSampleSize(FILEPATH, PARAMETERS, MEASURES, ATESTSIGLEVEL,
  ATESTRESULTSFILENAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS, TIMEPOINTS,
  TIMEPOINTS SCALE)
```

Finally, it may be interesting, especially for stochastic simulations, to see the distribution of simulation responses for each parameter value. A boxplot of these results can be produced for each parameter using the method below. Note that as this is produced directly from simulation responses, this plot can currently only be produced for simulation input provided in the first method, the organised folder structure.

```
oat_plotResultDistribution(FILEPATH, PARAMETERS, MEASURES, MEASURE_SCALE,
  CSV_FILE_NAME, BASELINE, PMIN, PMAX, PINC, PARAMVALS, TIMEPOINTS,
  TIMEPOINTS SCALE)
```

Similarly to Technique 1, it is possible to perform this technique for multiple simulation timepoints. To do this, the researcher needs to follow the same requirements detailed at the end of the description of Technique 1.

Running this analysis in R with the example data should produce the two A-Test graphs seen in Figures 2(B) and 2(C). Both are interesting for different reasons. In Figure 2(B), responses for both simulation measures greatly vary when the value of that parameter is perturbed, suggesting that assigning an appropriate value is critical. This change in behaviour can be compared with existing data, if applicable, or expert opinion, to determine if the parameter is having the desired effect. Yet in Figure 2(C), it appears that the simulation response does not significantly change for all parameter values that were explored, thus the simulation is robust to a change in parameter value. There is a large uncertainty in the value that should be assigned to this parameter. Reading of this result is dependent

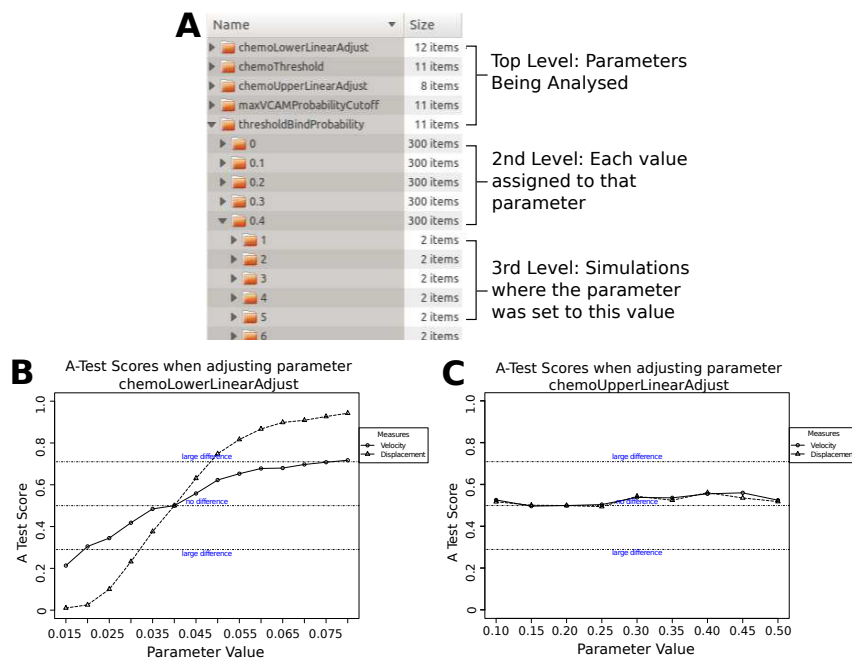


Figure 2: Use of **spartan** to determine how robust a simulation is to parameter perturbation. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: A-Test scores generated when the parameter controlling initial expression of a chemoattractant is perturbed. C: A-Test scores generated when the parameter controlling saturation of a chemoattractant is perturbed. B and C are reproduced from Alden et al. (2013b)

on the context of the work being performed, as this may suggest that assigning the parameter a correct value is not critical for simulation response, yet could also suggest that efforts should be made to obtain a value for this parameter using other means (i.e. lab experimentation).

This technique provides researchers with the ability to understand if the parameter has captured the required behaviour correctly, and how that factor alone impacts simulation response. As noted above the generated results can be interpreted a number of ways, depending on the context. If expert opinion, or biological data in our case, concerning the range of possible parameter values is available, it can be contrasted with analysis results to infer how well the simulation captures the biology. If this is not available, the analysis results can reveal the implications of not knowing the data: there is little concern if the parameter has little effect, otherwise efforts should be focused on careful calibration, or it can point to the need to further investigate the real world system.

Technique 3: Latin-hypercube analysis

Although parameter robustness is useful for establishing robustness to alteration in the value of a single parameter, the technique can not reveal any second-order effects that occur when two or more parameters are perturbed simultaneously. The effect one parameter has on simulation response may be highly dependent on the value of another. A global sensitivity analysis is required to examine this. In compiling **spartan** we have provided two previously described global sensitivity analysis techniques. This section describes the first of these, generating simulation parameter value sets using latin-hypercube sampling (Read et al., 2012; Marino et al., 2008; Saltelli et al., 2000). Through simultaneously perturbing all parameters, those that highly influence simulation behaviour can be revealed, in turn suggesting the pathways or mechanisms that have greatest impact on simulation response. This has the potential to provide a unique insight into the system being studied.

For each parameter of interest, a parameter value space is assigned. Simulation values for each parameter are then selected using a latin-hypercube, a way of sampling a large volume with minimal number of parameter value samples while also reducing any possible correlations across each generated parameter value set. Simulation runs are performed for all the generated points in the parameter space, and correlations between parameter values and corresponding simulation responses are calculated. Large correlations indicate influential parameters. Although each parameter is investigated in turn, the analysis is global as all parameter values are changing: the effect of influential parameters can transcend any parameter specific noise **spartan** includes methods to produce simulation parameter value sets for this statistical technique and to analyse the resultant simulation response under those

conditions.

Parameter sampling

spartan utilises the **lhs** R package to assist in the generation of simulation parameters using latin-hypercube sampling. This package can produce a number of sets of values, for a given number of parameters, either using a 'normal' algorithm that is fairly fast, or an 'optimal' sample that ensures the parameter space is fully explored. Although the latter offers that potential benefit, it can take a while to generate the parameter value sets. The researcher should choose a number of simulation parameter sets that should be generated from the hypercube. In our case study, we adopt the approach described in [Read et al. \(2012\)](#), where the parameter space is sampled 500 times to ensure adequate coverage across the range of values for each parameter. Each set of the 500 samples contains a value for the six parameters of interest. The complete set of hypercube samples is output as a CSV file. Simulations should now be performed under the conditions specified by each parameter set, each repeated for an appropriate number of runs (determined using Technique 1) where the simulation is not deterministic. In our case study, each of the 500 samples was run 300 times, to ensure the impact that inherent aleatory uncertainty has on simulation response is mitigated.

```
# Import the packages
library(spartan)
library(lhs)
# The folder where the parameter samples should be output to
FILEPATH <- "/home/user/LHC/Sampling"
# Names of the parameters to generate values for.
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold", "chemoUpperLinearAdjust",
  "chemoLowerLinearAdjust", "maxVCAMeffectProbabilityCutoff", "vcamSlope")
# The number of parameter sample sets to create using the hypercube
NUMSAMPLES <- 500
# The minimum value in the range for each parameter
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25)
# The maximum value in the range for each parameter
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0)
# Algorithm to use to generate the hypercube. Can be normal (quick) or optimal,
# which can take a long time (especially for high number of parameters)
ALGORITHM <- "normal"

lhc_generate_lhc_sample(FILEPATH, PARAMETERS, NUMSAMPLES, PMIN, PMAX, ALGORITHM)
```

Analysing the results

In this section we will demonstrate how **spartan** can produce statistical information that identifies any non-linear effects between parameters. Again we utilise our exemplar application and the example data available from our laboratory website (www.ycil.org.uk). Note that to reduce the size of the download data, the original simulation responses have not been included in the data: rather the summaries of the number of runs performed under each condition are provided. However, we do describe how these summaries are constructed below. The first step in this demonstration is to define the objects required for this analysis, as below. Again to save space, we have not duplicated the comments for objects that were declared in previous examples: the reader should refer to these if required.

```
library(spartan)
# Folder containing the example simulation results. Make sure the folder is unzipped
FILEPATH <- "/home/user/LHC/LHC_Results"
PARAMETERS <- c("thresholdBindProbability", "chemoThreshold", "chemoUpperLinearAdjust",
  "chemoLowerLinearAdjust", "maxVCAMeffectProbabilityCutoff", "vcamSlope")
MEASURES <- c("Velocity", "Displacement")
MEASURE_SCALE <- c("microns/min", "microns")
# Number of parameter value sets created in latin-hypercube sampling
NUMSAMPLES <- 500
# Number of simulation runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
RESULTSFILNAME <- "trackedCells_Close.csv"
ALTERNATIVEFILENAME <- NULL
```

```

OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
# This is either 1: The name of the single CSV file that summarises all simulation runs
# of all parameter sets generated by the hypercube (using input format 2), or the name
# to assign this file when spartan produces it from simulation results (using input
# format 1).
LHC_ALL_SIM_RESULTS_FILE <- "LHC_AllResults.csv"
# Location of a file containing the parameter value sets generated by the hypercube
# sampling (i.e. the file generated in the previous function of this paper). However
# if providing a CSV file with all results, you do not need to provide this
LHC_PARAM_CSV_LOCATION <- "Tutorial_Parameters_for_Runs.csv"
# spartan produces a summary file showing the parameter value sets alongside the
# median results for each simulation output measure. This names this file.
# Note no file extension
LHCSUMMARYFILENAME <- "LHC_Summary.csv"
# File name to give to the file showing the Partial Rank Correlation Coefficients
# for each parameter.
CORCOEFFSOUTPUTFILE <- "LHC_corCoeffs.csv"
# Timepoints being analysed. Must be NULL if no timepoints being analysed, or else
# be an array of timepoints. Scale sets the measure of these timepoints
TIMEPOINTS<-NULL; TIMEPOINTSCALE<-NULL
# Example Timepoints, if being used:
#TIMEPOINTS <- c(12, 36, 48, 60); TIMEPOINTSCALE <- "Hours"

```

In this example, we are analysing the result of a global sensitivity analysis for six parameters. The latin-hypercube has been used to produce 500 parameter value sets (NUMSAMPLES). Similarly to Techniques 1 and 2 above, this technique can process simulation responses in two formats. The first consists of an organised folder structure that contains CSV files containing the results of each simulation run (Figure 3(A)). The structure is comprised of two levels: the number of the latin-hypercube sample being analysed, containing simulation runs under those parameter conditions. However, if a researcher prefers, they can provide a CSV file summarising the simulation responses for all parameter sets generated by the hypercube. Each row of the CSV file should contain the parameter values selected from the hypercube and the median of each simulation response produced in that simulation run. For stochastic simulations where a number of runs are required to produce a consistent result (see Technique 1), a number of results for each parameter set should be provided in the CSV file. For example, in our case study we have repeated each simulation 300 times, thus 300 lines exist in the CSV file for each parameter value set generated by the hypercube.

It is this CSV file that the exemplar data from our website contains: the original simulation data has not been provided due to the large file size. However, this CSV file would be constructed using the method below. Similarly to Technique 2 above, the function processes each hypercube parameter value set in turn, producing a summary of the responses for each simulation run under those parameter value conditions. In the exemplar simulation data, it can be noted that multiple results have been included for each parameter set, to mitigate the impact of any aleatory uncertainty (Technique 1). With all parameter value sets processed, the CSV file is output with the filename stated in LHC_ALL_SIM_RESULTS_FILE.

```

lhc_process_sample_run_subsets(FILEPATH, LHC_PARAM_CSV_LOCATION, PARAMETERS, NUMSAMPLES,
  NUMRUNSPERSAMPLE, MEASURES, RESULTFILENAME, ALTERNATIVEFILENAME, OUTPUTCOLSTART,
  OUTPUTCOLEND, LHC_ALL_SIM_RESULTS_FILE)

```

With the CSV file in place (either constructed using the method above or provided), the next **spartan** function (below) summarises this further, generating the median of each simulation response for all runs under each hypercube parameter condition. The summary file is saved in the directory specified in FILEPATH, named as stated in LHCSUMMARYFILENAME.

```

lhc_generateLHCsummary(FILEPATH, PARAMETERS, MEASURES, LHC_ALL_SIM_RESULTS_FILE,
  LHCSUMMARYFILENAME, LHC_PARAM_CSV_LOCATION, TIMEPOINTS, TIMEPOINTSCALE)

```

With the parameter values and median of each simulation response together, **spartan** can now process this table. Each parameter is taken in turn, and the simulation responses ordered by the value assigned to that parameter. The objective is to identify any correlation between the value assigned to the parameter and the simulation response that is discernible despite the noise created by all other parameters varying randomly. Strong correlations correspond to influential parameters.

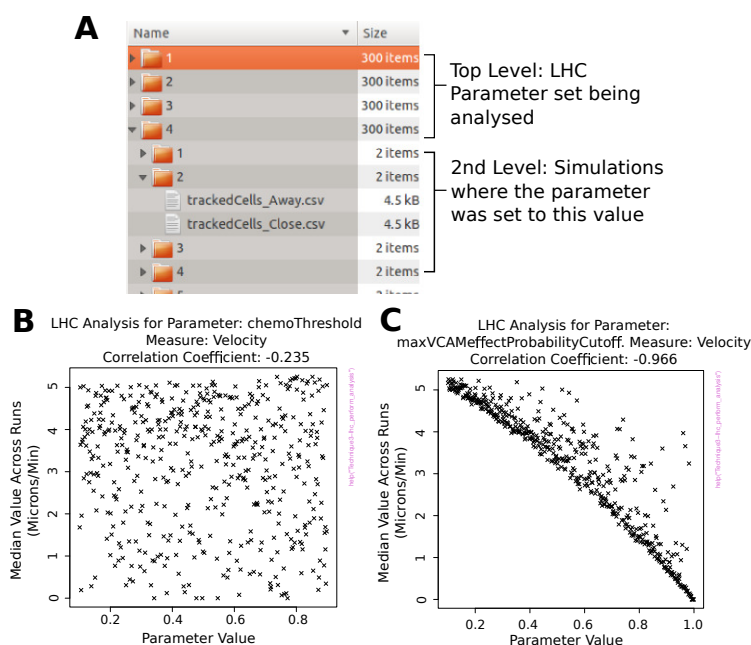


Figure 3: Use of **spartan** to identify compound effects between parameters using latin-hypercube sampling. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: Simulation responses sorted by the parameter that captures chemoattractant expression level required to influence cell motility, showing no trend between parameter value and result. C: Responses sorted by the parameter capturing the level of cell adhesion required to restrict cell motility, where a clear trend is apparent. B and C are reproduced from Alden et al. (2013b)

spartan provides two methods for helping to identify any correlations. The first is the calculation of Partial Rank Correlation Coefficients (PRCC) for each parameter. This statistical measure accounts for non-linear relationships between parameter and response, correcting for any effects by other parameters on the response. When the first of the below methods is run, these coefficients are calculated for each parameter, with the coefficients and p-values stored in a CSV file named as specified in CORCOEFFSOUTPUTFILE. The second method aids identification of any non-linear effects by producing a plot for each parameter, showing the simulation response observed for each value assigned to that parameter. Now any strong correlations, or any interesting areas where correlations appear at certain values, can be easily identified. The PRCC value is stated in the graph header, providing a statistical measure alongside the visual of simulation responses. Once the methods are run, these graphs will be found in the folder specified by FILEPATH.

```
lhc_generatePRCCoEffs(FILEPATH, PARAMETERS, MEASURES, LHCSUMMARYFILENAME,
CORCOEFFSOUTPUTFILE, TIMEPOINTS, TIMEPOINTSSCALE)
```

```
lhc_graphMeasuresForParameterChange(FILEPATH, PARAMETERS, MEASURES, MEASURE_SCALE,
CORCOEFFSOUTPUTFILE, LHCSUMMARYFILENAME, TIMEPOINTS, TIMEPOINTSSCALE)
```

Two of the plots produced for the example data can be seen in Figure 3(B) and 3(C). In Figure 3(B), no clear trend emerges for this parameter. In contrast, there is a strong correlation between the value of the parameter in Figure 3(C) and the simulation response (cell velocity). This suggests that, although a number of other parameters are also being perturbed, the value of this parameter has a significant impact on simulation response. Through this analysis, we identified that cell adhesion may be a key pathway at this point in lymphoid organ development, and thus is a pathway to be explored further in the laboratory. This demonstrates the impact that running a global sensitivity analysis could have on simulation response. With **spartan**, the researcher has the tools to perform both the parameter sampling and analysis of simulation data, to help draw such conclusions.

Similarly to the techniques above, it is possible to perform this technique for multiple simulation timepoints. To do this, the researcher needs to follow the same requirements detailed at the end of the description of Technique 1.

Technique 4: Extended fourier amplitude sampling test

The above global sensitivity analysis technique is termed a sampling-based technique. As an alternative, **spartan** includes a variance-based technique, the Extended Fourier Amplitude Sampling Test (eFAST) (Marino et al., 2008; Saltelli and Bollardo, 1998). Whereas Technique 3 looks for correlations between simulation response and value of a specific parameter, eFAST partitions simulation output variance that is caused by parameter perturbation between the input parameters, providing a statistical measure of the proportion of variance accounted for by each parameter of interest. This measure gives a strong indication of the influence of each parameter, and thus the influential pathways in the simulation.

Of the four techniques this is by far the most complex, and the reader is directed to our **spartan** publication (Alden et al., 2013b) and the available literature that describes eFAST in detail (Marino et al., 2008; Saltelli and Bollardo, 1998) for a comprehensive description. As an overview, a parameter space is set for each parameter and an additional parameter, the 'dummy'. The reasoning for the introduction of a dummy parameter that has no impact on simulation response is noted later. Taking each in turn as that of interest, values are chosen for all parameters through the use of sinusoidal functions of a particular frequency through the parameter space, with the frequency assigned to the parameter of interest significantly different from that assigned to the others being examined. From each curve, a number of parameter values are selected. As sinusoidal curves have symmetrical properties, there is the potential that the same value sets could be chosen more than once. This is avoided by the introduction of resample curves, that introduce a phase shift into each frequency. With this introduced, the sampling is performed again, and repeated for a set number of resample curves (or phase shifts). It is vital the reader appreciates the importance of setting the number of samples from each curve and the number of resample curves correctly, making themselves familiar with equations in Marino et al. (2008) that aid this decision. From this process, the researcher creates a number of value sets for each parameter, for each curve. Where there are a large number of parameters in the analysis, this can lead to a large number of parameter value sets under which conditions simulations should be run. Thus this technique can be computationally expensive (Tarantola et al., 2006), especially for stochastic simulations where these runs need to be repeated.

The analysis of the simulation responses takes the frequency used to generate the parameter set into account. Utilising Fourier analysis, variation in the simulation response can be partitioned between the parameters under study. For each parameter, two statistical measures are calculated: the fraction of output variance that can be explained by the value the parameter has been assigned (S_i), and the fraction of variance in the response due to higher-order non-linear affects between other simulation parameters and the parameter of interest. A parameter is deemed to have a statistically significant impact on simulation response when the sensitivity measures S_i and ST_i are contrasted to those calculated for the dummy parameter, using a two-sample t-test.

Parameter sampling

Generation of parameter samples using the eFAST technique noted above has been greatly aided by Marino et al. (2008) making their MATLAB code available, easing translation of this technique into R. The researcher should ensure they have read the available literature for this technique to ensure the number of curves and value samples taken from these curves is chosen appropriately (Alden et al., 2013b; Marino et al., 2008; Saltelli and Bollardo, 1998). As described in brief above, the analysis focuses on each parameter at a time, for each curve. Thus, when the below function is run, **spartan** produces a CSV file for each parameter, for each resample curve. In the example below, we have 7 parameters (6 plus the 'dummy') and 3 resample curves: leading **spartan** to produce 21 CSV files. Simulation runs should be performed for each parameter value set in these files: 1365 sets in the case of this example. It is easy to note how this technique becomes computationally expensive as parameter numbers increase, especially for stochastic simulations.

```
# Import the package
library(spartan)
# The folder where the parameter samples should be output to
FILEPATH <- "/home/user/eFAST/"
# Number of resample curves (phase shifts) to use in the sampling procedure
NUMCURVES <- 3
# Names of the parameters to generate parameter value samples for.
PARAMETERS <- c("BindProbability", "ChemoThreshold", "ChemoUpperLinearAdjust",
                "ChemoLowerLinearAdjust", "VCAMProbabilityThreshold", "VCAMSlope", "Dummy")
# The number of parameter sample sets to create for each curve
NUMSAMPLES <- 65
```



```
# The minimum value in the range for each parameter
PMIN <- c(0, 0.10, 0.10, 0.015, 0.1, 0.25, 1)
# The maximum value in the range for each parameter
PMAX <- c(100, 0.9, 0.50, 0.08, 1.0, 5.0, 10)

efast_generate_sample(FILEPATH, NUMCURVES, NUMSAMPLES, PARAMETERS, PMIN, PMAX)
```

Analysing the results

Here we utilise our example simulation data to demonstrate the application of the eFAST technique available in **spartan**. The first step in this demonstration is to define the objects required for this analysis, as below. Again to save space, we have not duplicated the comments for objects that were declared in previous examples: the reader should refer to these above if necessary. To aid graphing of the results generated by this analysis, we utilise the **gplots** R package.

```
library(spartan)
library(gplots)
# Folder containing the eFAST simulation results. Make sure example data is unzipped
FILEPATH <- "/home/user/eFAST/Results"
PARAMETERS <- c("BindProbability", "ChemoThreshold", "ChemoUpperLinearAdjust",
               "ChemoLowerLinearAdjust", "VCAMProbabilityThreshold", "VCAMSlope", "Dummy")
MEASURES <- c("Velocity", "Displacement")
RESULTFILENAME <- "trackedCells_Close.csv"
ALTERNATIVEFILENAME <- NULL
OUTPUTCOLSTART <- 10
OUTPUTCOLEND <- 11
# Number of resample curves employed when the parameter space was sampled
NUMCURVES <- 3
# The number of parameter sample sets taken from each curve
NUMSAMPLES <- 65
# Number of simulation runs performed for each parameter value set
NUMRUNSPERSAMPLE <- 300
# Which of the output measures to T-Test for significance
OUTPUTMEASURES_TO_TTEST <- 1:2
# T-Test confidence level
TTEST_CONF_INT <- 0.95
# Name of the final result file for this analysis, showing the partitioning of
# the variance between input parameters
EFASTRESULTFILENAME <- "EgSet_eFAST_Analysis.csv"
# Boolean to note whether summary graphs should be produced
GRAPH_FLAG <- TRUE
TIMEPOINTS <- NULL; TIMEPOINTSCALE <- NULL
```

In this example, we are utilising Fourier frequency analysis to examine the results of perturbing six parameters simultaneously (7 including the dummy). Similarly to all the previous techniques described in this paper, the researcher can specify their simulation result input in two formats. The first consists of the organised folder structure described in Figure 4(A). The structure is comprised of four levels: the sample curve number, then parameter of interest for that curve, containing the number of the set of simulator values generated for that parameter, that in turn holds the responses for simulations run under those conditions. As noted previously, such a structure is suitable in cases where a large number of simulation runs have been performed, usually on a computer cluster. This technique is one of these cases, as the large number of parameter value sets generated requires a large number of simulation results. However, the researcher can also provide their input in a series of CSV files, one per curve-parameter pairing. Each row of this file should contain the parameter set generated from the sinusoidal curve and the median simulation response when a run was performed under those conditions. Multiple rows can exist per parameter set, representing multiple runs performed to mitigate aleatory uncertainty. If the researcher chooses this method, they should follow the convention of naming the file *Curve[Curve Number]_Parameter[Parameter Number]_Results.csv*, for example *Curve1_Parameter1_Results.csv*.

The exemplar simulation data contains the latter: the summary CSV files for each curve-parameter pairing. With this analysis requiring a large number of simulation runs, it has not been possible to provide all the simulation runs in an appropriate size download. Similarly to the previously described

techniques, **spartan** processes the runs in these folders to create a CSV file summarising the simulation runs: one per curve-parameter pair, formatted as described above. In our case study, where we have multiple values for each response (i.e. cells), the median of these responses is calculated and stored in the summary file, for each run. Where the responses for all simulation runs are available, the following function can be used to produce these files:

```
efast_generate_medians_for_all_parameter_subsets(FILEPATH, NUMCURVES, PARAMETERS,
        NUMSAMPLES, NUMRUNSPERSAMPLE, MEASURES, RESULTFILENAME, ALTERNATIVEFILENAME,
        OUTPUTCOLSTART, OUTPUTCOLEND, TIMEPOINTS, TIMEPOINTSSCALE)
```

With the CSV file summaries in place (either provided by the researcher or generated using the function above), the variance in simulation response can be partitioned between the parameters. To do this, the algorithm needs to generate a further summary of all simulation responses: one for each parameter resampling curve employed. This is generated by the following function. A CSV summary file is produced that shows, for each parameter of interest, the median of each simulation output response under the generated parameter value set conditions. As an example, consider we are examining curve 1, parameter 1. For this parameter of interest, 65 different parameter value sets were generated from the frequency curves, thus we have 65 different sets of simulation results. The previous method produced a summary of the responses for each simulation run under those conditions: this method now takes the median of each response and stores this in the summary alongside the parameter values that generated these medians. Thus, for each parameter of interest, median responses for each of the 65 sets of results are stored. The next parameter is then examined, until all have been analysed. This produces a snapshot showing the median simulation output for all parameter value sets generated for the first resample curve. This is stored in the directory specified by the FILEPATH object, with the name Curve[Curve Number]_Summary.csv.

```
efast_get_overall_medians(FILEPATH, NUMCURVES, PARAMETERS, NUMSAMPLES, MEASURES,
        TIMEPOINTS, TIMEPOINTSSCALE)
```

It is this summary that can then be processed by the eFAST algorithm, producing statistical measures that partition the variance in simulation response between the parameters of interest:

```
efast_run_Analysis(FILEPATH, MEASURES, PARAMETERS, NUMCURVES, NUMSAMPLES,
        OUTPUTMEASURES_TO_TTEST, TTEST_CONF_INT, GRAPH_FLAG, EFASTRESULTFILENAME,
        TIMEPOINTS, TIMEPOINTSSCALE)
```

In the folder specified in FILEPATH, an overall summary of the analysis will have been produced, named as stated in EFASTRESULTFILENAME. The summary file contains a row showing the name of each parameter being examined, followed by a number of statistical measures generated for each simulation response: the fraction of output variance explained by a variation of that parameter (S_i); the fraction of variance accounted for by that parameter and any higher-order or non-linear effects between the parameter and others (ST_i); the variance caused by all parameters excluding that of interest (SC_i); p-values for S_i and ST_i when contrasted with those of the dummy parameter using the two-sample t-test; Standard Error values, which can be calculated as there are a number of resample curves. For ease of representation, a graph is produced for each simulation output response showing the S_i and ST_i values: as demonstrated in Figure 4(B). A parameter is deemed to have a significant impact on simulation response when the S_i and ST_i statistical measures are contrasted with those for the dummy parameter, known to have no effect on simulation response. In terms of the example data, it is clearly apparent from the graph that one parameter has a highly significant effect on the cell velocity simulation response. Two-sample t-test comparisons however suggest there are four parameters that are statistically significant in contrast with the dummy parameter, with no significant role for the pathways represented by the remaining two parameters.

Again it is possible to perform this technique for multiple simulation timepoints, if the requirements detailed at the end of the description of Technique 1 are followed.

Although this is a complex statistical technique, gaining a statistical measure of the impact of each parameter on simulation variance is useful in fully understanding simulator behaviour, especially if this conclusion is supported by other techniques above. Unless the reader has significant computational resources, this technique is more powerful for analyses containing just a few parameters.

Conclusion

Here we have presented an exemplar application of the statistical techniques available in **spartan**. The four techniques that we have compiled within the package have proved very fruitful in suggesting

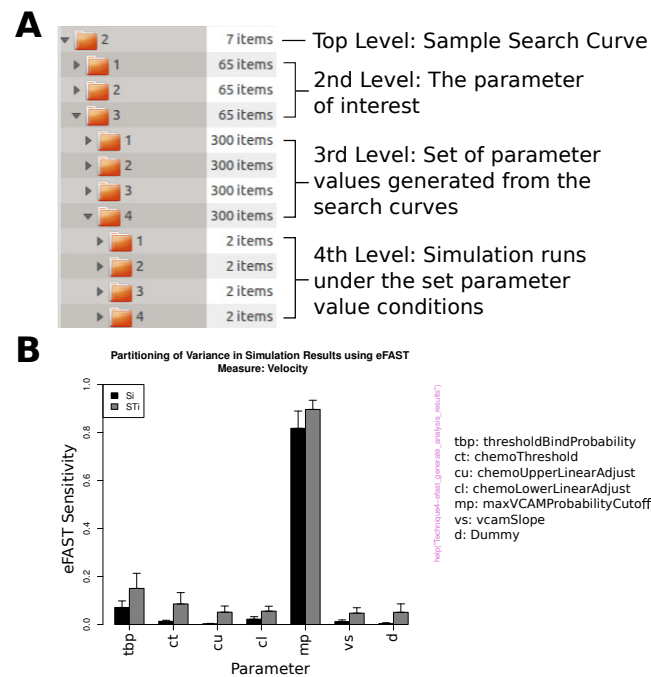


Figure 4: Use of eFAST in partitioning variance in response between parameters. A: The folder structure that should be utilised if providing all simulation result files rather than a CSV file summarising all results. B: Sensitivity measures for the cell velocity response. Black bars: Si - variance explained by the value assigned to that parameter. Grey bars: STi: Variance caused by this parameter and non-linear effects between this parameter and others. B is reproduced from Alden et al. (2013b).

potential biological pathways that should be explored further in attempts to understand the development of secondary lymphoid tissue. Through providing this demonstration, example simulation data, and the accompanying **spartan** publication, we encourage researchers across disciplines to utilise the toolkit to increase confidence in their generated simulator. **spartan** can produce statistical analyses to accompany simulation results, revealing that the researcher has considered the effect that any inherent stochasticity has on their results, that they understand how robust their simulator is to parameter perturbation, and that they understand the key pathways or mechanisms identified through global sensitivity analysis.

The scope for applying the techniques available is much wider than agent-based models of biological systems: capacity exists to utilise **spartan** alongside traditional and partial differential equation models as well as those where an agent-based approach has been adopted. In these scenarios the researcher would not need to perform the first function of all the techniques described above, where the results from a number of runs under the same parameter condition are summarised to take stochastic variation into account. Instead, single simulation results are placed into the same file structure described in each method, and the **spartan** analysis run as described above.

Acknowledgements

This work was part-funded by the Wellcome Trust [ref:097829] through the Centre for Chronic Diseases and Disorders (C2D2) at the University of York. Jon Timmis is part funded by the Royal Society and The Royal Academy of Engineering. Mark Read was funded by FP7: ICT grant GA 270382 "Collective Cognitive Robotics," Paul Andrews was funded by EPSRC grant EP/I005943/1 "Resilient Futures."

Bibliography

- K. Alden, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles. Pairing experimentation and computational modelling to understand the role of tissue inducer cells in the development of lymphoid organs. *Frontiers in Immunology*, 3:1–20, 2012. [p63, 64, 69]
- K. Alden, M. Read, P. Andrews, J. Timmis, H. Veiga-Fernandes, and M. Coles. *spartan: Spartan (Simulation Parameter Analysis R Toolkit Application)*, 2013a. URL <http://CRAN.R-project.org/package=spartan>. R package version 2.1. [p64]

- K. Alden, M. Read, J. Timmis, P. S. Andrews, H. Veiga-Fernandes, and M. C. Coles. Spartan: A comprehensive tool for understanding uncertainty in simulations of biological systems. *PLoS Computational Biology*, 9(2), 2013b. [p63, 64, 67, 71, 74, 75, 78]
- R. Carnell. *lhs: Latin hypercube samples*, 2012. URL <http://CRAN.R-project.org/package=lhs>. R package version 0.10. [p64]
- G. M. Dancik. *mleqp: Maximum likelihood estimates of gaussian processes*, 2013. URL <http://CRAN.R-project.org/package=mleqp>. R package version 3.1.4. [p63]
- J. C. Helton. Uncertainty and sensitivity analysis for models of complex systems. In T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, editors, *Computational Methods in Transport: Verification and Validation*, pages 207–228. Springer, 2008. [p64]
- M. Ivanyi, R. Bocsi, L. Gulyas, V. Kozma, and R. Legendi. The multi-agent simulation suite. In *AAAI Fall Symposium Series*, pages 56–63, 2007. [p63]
- M. Lamboni, H. Monod, and C. Bidot. *multisensi: Multivariate sensitivity analysis*, 2013. URL <http://CRAN.R-project.org/package=multisensi>. R package version 1.0.7. [p63]
- D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2013. URL <http://CRAN.R-project.org/package=XML>. R package version 3.98-1.1. [p64]
- S. Luke. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005. [p63]
- S. Marino, I. B. Hogue, C. J. Ray, and D. E. Kirschner. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of Theoretical Biology*, 254(1):178–96, 2008. [p63, 71, 75]
- M. Meier-Schellersheim, X. Xu, B. Angermann, E. J. Kunkel, T. Jin, and R. N. Germain. Key role of local regulation in chemosensing revealed by a new molecular interaction-based modeling method. *PLoS Computational Biology*, 2(7):710–724, 2006. [p63]
- A. Patel, N. Harker, L. Moreira-Santos, M. Ferreira, K. Alden, J. Timmis, K. E. Foster, A. Garefalaki, P. Pachnis, P. S. Andrews, H. Enomoto, J. Milbrandt, V. Pachnis, M. C. Coles, D. Kioussis, and H. Veiga-Fernandes. Differential RET responses orchestrate lymphoid and nervous enteric system development. *Science Signalling*, 5(235), 2012. [p63, 64]
- G. Pujol, B. Iooss, A. Janon, P. Lemaitre, L. Gilquin, L. L. Gratiet, T. Touati, B. Ramos, J. Fruth, and S. D. Veiga. *sensitivity: Sensitivity Analysis*, 2014. URL <http://CRAN.R-project.org/package=sensitivity>. R package version 1.9. [p63]
- M. Read, P. S. Andrews, J. Timmis, and V. Kumar. Techniques for grounding agent-based simulations in the real domain : a case study in Experimental Autoimmune Encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):67–86, 2012. [p63, 64, 66, 68, 71, 72]
- A. Saltelli and R. Bollardo. An alternative way to compute Fourier amplitude sensitivity test (FAST). *Comput. Stat. Data Anal.*, 26(4):445–460, 1998. [p63, 75]
- A. Saltelli, K. Chan, and E. M. Scott. *Sensitivity analysis*. Wiley series in probability and statistics Wiley, 2000. [p71]
- S. Tarantola, D. Gatelli, and T. Mara. Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering & System Safety*, 91(6):717–727, 2006. [p75]
- A. Vargha and H. D. Delaney. A critique and improvement of the cl common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioural Statistics*, 25:101–132, 2000. [p64, 68]
- G. R. Warnes, B. Bolker, L. Bonebakker, R. Gentleman, W. H. A. Liaw, T. Lumley, M. Maechler, A. Magnusson, S. Moeller, M. Schwartz, and B. Venables. *gplots: Various R programming tools for plotting data*, 2013. URL <http://CRAN.R-project.org/package=gplots>. R package version 2.12.1. [p64]
- U. Wilensky. *NetLogo itself*. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999. URL <http://ccl.northwestern.edu/netlogo/>. [p63]

Kieran Alden

*York Computational Immunology Laboratory, Centre for Immunology & Infection, and Dept of Electronics
University of York, Heslington, York. YO10 5DD
UK kieran.alden@york.ac.uk*

Mark Read

*York Computational Immunology Laboratory and Dept of Electronics
University of York, Heslington, York. YO10 5DD
UK mark.read@sydney.edu.au*

Paul Andrews

*York Computational Immunology Laboratory and Dept of Computer Science
University of York, Heslington, York. YO10 5DD
UK paul.andrews@york.ac.uk*

Jon Timmis

*York Computational Immunology Laboratory and Dept of Electronics
University of York, Heslington, York. YO10 5DD
UK jon.timmis@york.ac.uk*

Mark Coles

*York Computational Immunology Laboratory and Centre for Immunology & Infection
University of York, Heslington, York. YO10 5DD
UK mark.coles@york.ac.uk*

ngspatial: A Package for Fitting the Centered Autologistic and Sparse Spatial Generalized Linear Mixed Models for Areal Data

by John Hughes

Abstract Two important recent advances in areal modeling are the centered autologistic model and the sparse spatial generalized linear mixed model (SGLMM), both of which are reparameterizations of traditional models. The reparameterizations improve regression inference by alleviating spatial confounding, and the sparse SGLMM also greatly speeds computing by reducing the dimension of the spatial random effects. Package **ngspatial** ('ng' = non-Gaussian) provides routines for fitting these new models. The package supports composite likelihood and Bayesian inference for the centered autologistic model, and Bayesian inference for the sparse SGLMM.

Background and introduction

The traditional autologistic model (Besag, 1972) and areal GLMM (Besag et al., 1991) have enjoyed widespread popularity: they have been applied thousands of times in many fields, e.g., epidemiology, marketing, agriculture, ecology, forestry, geography, and image analysis. But it was recently discovered that both models are spatially confounded (Caragea and Kaiser, 2009; Clayton et al., 1993). This confounding can cause bias and/or variance inflation in the estimators of regression coefficients, leading to erroneous regression inference. This is a serious drawback because many spatial modelers are interested in regression (rather than, or in addition to, prediction).

To address the confounding of the traditional autologistic model, Caragea and Kaiser (2009) devised the centered autologistic model, so named because it replaces the traditional model's autocovariate with a centered autocovariate (see below for details).

The confounding of the mixed model was first addressed by Reich et al. (2006) using a technique now known as restricted spatial regression (Hodges and Reich, 2010). This technique alleviates spatial confounding and also yields a faster mixing Markov chain, but the computational burden remains high because the dimension of the spatial random effects is reduced only slightly relative to the traditional model. By using the so called Moran operator, Hughes and Haran (2013) were able to reparameterize the mixed model in a way that not only improves regression inference but also dramatically reduces the dimension of the random effects. The resulting model, which we will call the sparse SGLMM, can be fitted so efficiently that even the largest areal datasets can be analyzed quickly.

These promising new models cannot be applied using existing software, and so we have provided support for the models in version 1.0 of R package **ngspatial**, the subject of this article. First we discuss the two models in some detail. Then we present **ngspatial** 1.0, which permits composite likelihood and Bayesian inference for the centered autologistic model, and Bayesian inference for the sparse SGLMM. We conclude with a summary.

The models supported by ngspatial 1.0

Areal models

The autologistic model and the sparse SGLMM are areal models, i.e., models for data observed at the vertices of a graph $G = (V, E)$, where $V = \{1, 2, \dots, n\}$ are the vertices, and $E \subset V \times V$ are the edges. Each vertex of G represents an areal unit, i.e., an area over which measurements have been aggregated. For example, areal units could be counties, census tracts, voxels, pixels, or provinces. An edge (i, j) of G represents the spatial adjacency of areal units i and j . Typically, two areal units are considered adjacent if they share a boundary, but other definitions are possible. We will assume that G is undirected and free of loops and parallel edges.

The centered autologistic model

The traditional autologistic model was proposed by Besag (1972). The model is a Markov random field (MRF) model (Kindermann and Snell, 1980), which is to say that G describes conditional independencies among the random variables Z_i ($i = 1, \dots, n$) associated with V . For the autologistic model, the i th observation is Bernoulli distributed conditional on its neighbors:

$$\log \frac{\mathbb{P}(Z_i = 1 \mid \{Z_j : (i, j) \in E\})}{\mathbb{P}(Z_i = 0 \mid \{Z_j : (i, j) \in E\})} = \mathbf{x}'_i \boldsymbol{\beta} + \eta \sum_{j:(i,j) \in E} Z_j,$$

where \mathbf{x}_i is a p -vector of spatial predictors associated with the i th areal unit, $\boldsymbol{\beta}$ is a p -vector of spatial regression coefficients, η is a spatial dependence parameter, and $\sum Z_j$ is the so called autocovariate.

We see that η is a measure of Z_i 's reactivity to its neighbors. If $\eta = 0$, the model reduces to the ordinary Bernoulli GLM, while $\eta > 0$ (< 0) corresponds to positive (negative) spatial dependence. We will assume that $\eta > 0$ since the model is usually applied to phenomena that exhibit spatial attraction rather than repulsion.

Caragea and Kaiser (2009) showed that the traditional autologistic model is confounded. This is because the traditional autocovariate is not well suited to the task of fitting small-scale structure in the data, i.e., clustering induced by spatial dependence and residual to the large-scale structure $\mathbf{x}'\boldsymbol{\beta}$. Instead, the traditional autocovariate and the spatial predictors "compete" to explain the data, which prevents the model from isolating the role of the spatial predictors.

Caragea and Kaiser (2009) reparameterized the model by centering the autocovariate. The resulting conditional log odds are

$$\log \frac{\mathbb{P}(Z_i = 1 \mid \{Z_j : (i, j) \in E\})}{\mathbb{P}(Z_i = 0 \mid \{Z_j : (i, j) \in E\})} = \mathbf{x}'_i \boldsymbol{\beta} + \eta \sum_{j:(i,j) \in E} (Z_j - \mu_j),$$

where $\mu_j = \{1 + \exp(-\mathbf{x}'_j \boldsymbol{\beta})\}^{-1}$ is the independence expectation of Z_j . Centering allows the autocovariate to fit only residual structure so that fitting the large-scale structure is left to the regression term. Thus the centered model restores to $\boldsymbol{\beta}$ and η their desired interpretations as regression coefficients and dependence parameter, respectively.

Maximum likelihood and Bayesian inference for the autologistic model are complicated by an intractable normalizing function. To see this, first let $\mathbf{Z} = (Z_1, \dots, Z_n)'$; let \mathbf{X} be the design matrix; let $\mathbf{A} = [1\{(i, j) \in E\}]$ be the adjacency matrix of G , where $1\{\cdot\}$ is the indicator function; let $\boldsymbol{\theta} = (\boldsymbol{\beta}', \eta)'$ be the full parameter vector; and let $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)'$ be the vector of independence expectations. Then, assuming G has clique number 2, the joint pmf of the centered model is

$$\pi(\mathbf{Z} \mid \boldsymbol{\theta}) = c(\boldsymbol{\theta})^{-1} \exp \left(\mathbf{Z}' \mathbf{X} \boldsymbol{\beta} - \eta \mathbf{Z}' \mathbf{A} \boldsymbol{\mu} + \frac{\eta}{2} \mathbf{Z}' \mathbf{A} \mathbf{Z} \right), \tag{1}$$

where

$$c(\boldsymbol{\theta}) = \sum_{\mathbf{Y} \in \{0,1\}^n} \exp \left(\mathbf{Y}' \mathbf{X} \boldsymbol{\beta} - \eta \mathbf{Y}' \mathbf{A} \boldsymbol{\mu} + \frac{\eta}{2} \mathbf{Y}' \mathbf{A} \mathbf{Y} \right)$$

is the normalizing function (Hughes et al., 2011).

The normalizing function is intractable for all but the smallest datasets because the sample space $\{0, 1\}^n$ contains 2^n points. Our package offers two ways to solve this problem: (1) composite likelihood inference, which sidesteps $c(\boldsymbol{\theta})$, and (2) auxiliary-variable MCMC for Bayesian inference, which allows $c(\boldsymbol{\theta})$ to cancel from the Metropolis-Hastings acceptance probability. See below for details.

The sparse spatial generalized linear mixed model

The traditional SGLMM for areal data—sometimes referred to as the BYM model—was proposed by Besag, York, and Mollié (1991). The BYM model is hierarchical, inducing spatial dependence by way of a latent autonormal random vector. Conditional on these spatial random effects, the observations are independent and follow an ordinary GLM. Specifically, the transformed conditional means are

$$g(\mu_i) = \mathbf{x}'_i \boldsymbol{\beta} + S_i, \tag{2}$$

where g is a link function, $\mu_i = \mathbb{E}(Z_i \mid S_i)$, and S_i is the spatial random effect for the i th areal unit.

The most common specification for $\mathbf{S} = (S_1, \dots, S_n)'$ is the so called intrinsic conditional autoregression (ICAR), a zero-mean Gaussian Markov random field (GMRF; Rue and Held, 2005) with a

singular precision matrix that corresponds to the intuitively appealing full conditional distributions

$$S_i | \{S_j : (i, j) \in E\} \sim \mathcal{N} \left(\frac{1}{d_i} \sum_{j:(i,j) \in E} S_j, \frac{1}{\tau d_i} \right),$$

where d_i is the degree of vertex i and $\tau > 0$ is a smoothing parameter. These conditionals correspond to the joint distribution

$$S \sim \mathcal{N}\{\mathbf{0}, (\tau \mathbf{Q})^{-1}\},$$

where $\mathbf{Q} = \mathbf{D} - \mathbf{A}$, with $\mathbf{D} = \text{diag}(d_i)$. Since \mathbf{Q} is singular, the BYM model is employed from the Bayesian point of view, with the ICAR a prior distribution on S . Packages **CARBayes** (Lee, 2013) and **spdep** (Bivand, 2015) provide tools for fitting the ICAR and other CAR models.

Reich et al. (2006) showed that the BYM model is spatially confounded in the sense that the random effects can “pollute” the regression manifold $C(\mathbf{X})$, which can lead to a biased and variance-inflated posterior for β . To see this, first let \mathbf{P} be the orthogonal projection onto $C(\mathbf{X})$, so that $\mathbf{I} - \mathbf{P}$ is the orthogonal projection onto $C(\mathbf{X})^\perp$. Now eigendecompose \mathbf{P} and $\mathbf{I} - \mathbf{P}$ to obtain orthogonal bases ($\mathbf{K}_{n \times p}$ and $\mathbf{L}_{n \times (n-p)}$, say) for $C(\mathbf{X})$ and $C(\mathbf{X})^\perp$. Then (2) can be rewritten as

$$g(\mu_i) = \mathbf{x}'_i \beta + \mathbf{k}'_i \gamma + \mathbf{l}'_i \delta,$$

where $\gamma_{p \times 1}$ and $\delta_{(n-p) \times 1}$ are random coefficients. This form shows that \mathbf{K} is the source of the confounding, for \mathbf{K} and \mathbf{X} have the same column space.

Since the columns of \mathbf{K} are merely synthetic predictors (i.e., they have no scientific meaning), Reich et al. (2006) recommend removing them from the model. The resulting model (henceforth the RHZ model) has

$$g(\mu_i) = \mathbf{x}'_i \beta + \mathbf{l}'_i \delta,$$

so that spatial smoothing is restricted to the orthogonal complement of $C(\mathbf{X})$. In a subsequent paper, Hodges and Reich (2010) referred to this technique as restricted spatial regression.

Restricted spatial regression is not only an effective remedy for confounding but also speeds computing. Because the columns of \mathbf{L} are orthogonal, the RHZ model’s random effects are approximately *a posteriori* uncorrelated. This yields a fast mixing Markov chain, and the cost per iteration is reduced because a simple spherical normal proposal is sufficient for updating the random effects. But fitting the RHZ model to large areal datasets is still quite burdensome computationally because the random effects remain high dimensional.

By taking full advantage of G , Hughes and Haran (2013) were able to greatly reduce the number of random effects while also improving regression inference. Hughes and Haran (2013) begin by defining the so called Moran operator for \mathbf{X} with respect to G : $(\mathbf{I} - \mathbf{P})\mathbf{A}(\mathbf{I} - \mathbf{P})$. This operator appears in the numerator of a generalized form of Moran’s I , a popular nonparametric measure of spatial dependence (Moran, 1950):

$$I_{\mathbf{X}}(\mathbf{A}) = \frac{n}{\mathbf{1}'\mathbf{A}\mathbf{1}} \frac{\mathbf{Z}'(\mathbf{I} - \mathbf{P})\mathbf{A}(\mathbf{I} - \mathbf{P})\mathbf{Z}}{\mathbf{Z}'(\mathbf{I} - \mathbf{P})\mathbf{Z}}.$$

Boots and Tiefelsdorf (2000) showed that (1) the (standardized) spectrum of a Moran operator comprises the possible values for the corresponding $I_{\mathbf{X}}(\mathbf{A})$, and (2) the eigenvectors comprise all possible mutually distinct patterns of clustering residual to $C(\mathbf{X})$ and accounting for G . The positive (negative) eigenvalues correspond to varying degrees of positive (negative) spatial dependence, and the eigenvectors associated with a given eigenvalue (λ_i , say) are the patterns of spatial clustering that data exhibit when the dependence among them is of degree λ_i .

In other words, the eigenvectors of the Moran operator form a multiresolutional spatial basis for $C(\mathbf{X})^\perp$ that exhausts all possible patterns that can arise on G . Since we do not expect to observe repulsion in the phenomena to which these models are usually applied, we can use the spectrum of the operator to discard all repulsive patterns, retaining only attractive patterns for our analysis (although it can be advantageous to accommodate repulsion (Griffith, 2006)). The nature of the attractive Moran eigenvectors is illustrated in Figure 1.

By retaining only eigenvectors that exhibit positive spatial dependence, we can usually reduce the model dimension by at least half *a priori*. And Hughes and Haran (2013) showed that a much greater reduction is possible in practice, with 50–100 eigenvectors being sufficient for most datasets.

Let $\mathbf{M}_{n \times q}$ contain the first $q \ll n$ eigenvectors of the Moran operator. Then the sparse SGLMM has first stage

$$g(\mu_i) = \mathbf{x}'_i \beta + \mathbf{m}'_i \delta_s,$$

where \mathbf{m}_i is the i th row of \mathbf{M} and δ_s is a q -vector of random coefficients. This implies $p + q + 1$ model

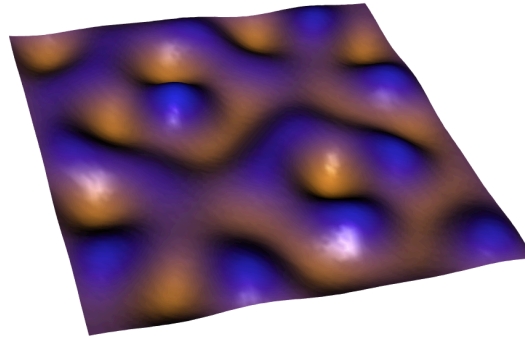


Figure 1: A Moran eigenvector exhibiting strong positive spatial dependence. The graph is the square 100×100 lattice. The coordinates of the vertices ($\mathbf{x}_i = (x_i, y_i)'$ for $i = 1, \dots, 10\,000$) were restricted to the unit square, and those coordinates were used as spatial covariates, i.e., $\mathbf{X}' = (\mathbf{x}_1 \cdots \mathbf{x}_{10\,000})$.

parameters, compared to $p + n + 1$ for the traditional model and $p + (n - p) + 1 = n + 1$ for the RHZ model. This dramatic reduction in dimension speeds computation considerably, allowing even the largest areal datasets to be analyzed quickly.

We note that the sparse SGLMM was partly inspired by spatial filtering, also known as principle coordinates of neighbor matrices. See, e.g., Griffith (2003); Dray et al. (2006); Borcard et al. (2011) for more information. Package **spdep** provides tools for spatial filtering.

Package **ngspatial**

Package **ngspatial** supports composite likelihood and Bayesian inference for the centered autologistic model, and Bayesian inference for the sparse SGLMM. Usage of the package's two main functions, `autologistic` and `sparse.sglm`, is simple and intuitive while offering considerable flexibility.

Fitting the centered autologistic model

The fitting function for the centered autologistic model is called `autologistic`. This function returns an object of class "autologistic". Auxiliary functions `residuals`, `summary`, and `vcov` accept an object of type "autologistic" and return various kinds of residuals, summarize the fit, and return an estimated covariance matrix, respectively.

A fast perfect sampler for the centered autologistic model

A key component of our autologistic framework is function `rautologistic`, which simulates a draw from a centered autologistic model. The function has the following signature.

```
rautologistic(X, A, theta)
```

The three arguments are as described above. The functions of **ngspatial** require that an adjacency matrix be binary and symmetric (see function `isSymmetric` from the **base** package) and of type `matrix`. Note that package **spdep** provides function `nb2mat`, which constructs an adjacency matrix from a neighbors list. **spdep** also provides many other functions that are useful for handling adjacency structures.

This function employs coupling from the past (Propp and Wilson, 1996) to generate a vector distributed exactly according to (1). We use perfect sampling, rather than ordinary MCMC, for two reasons. First, the MCMC algorithm we use to do Bayesian inference requires that we draw a perfect sample during each iteration (Møller et al., 2006). Were we to use an approximation, the resulting Markov chain would not necessarily have the true posterior as its stationary distribution (Murray et al., 2006). Second, although perfect sampling can be computationally burdensome, a carefully implemented perfect sampler is fast enough to permit composite likelihood analysis of even very large datasets, while obviating convergence diagnosis.

Our perfect sampler was implemented in C++. More specifically, we used the Armadillo linear algebra library (Sanderson, 2010), which provides two benefits: (1) intuitive syntax for linear algebra (e.g., $\mathbf{Z}'\mathbf{X}\beta$ can be computed as `Z.t() * X * beta`), and (2) speed (Armadillo uses delayed evaluation to combine multiple operations, thus reducing or eliminating the need for temporaries). We integrated the C++ and R code using the **Rcpp** and **RcppArmadillo** packages (Eddelbuettel and François, 2011).

We tested our Armadillo-based sampler in a number of circumstances and found it to be over three times as fast as an optimal pure R version.

Composite likelihood inference for the centered autologistic model

One way to overcome the intractability of the normalizing function $c(\boldsymbol{\theta})$ is to avoid it. This can be accomplished by considering the so called pseudolikelihood (PL), which is a composite likelihood (Lindsay, 1988) of the conditional type. Each of the n factors in the pseudolikelihood is the likelihood of a single observation, conditional on said observation’s neighbors:

$$p_i(\boldsymbol{\theta})^{z_i} \{1 - p_i(\boldsymbol{\theta})\}^{1-z_i} = \mathbb{P}(Z_i = z_i \mid \{Z_j : (i, j) \in E\}) = \frac{\exp[z_i \{ \mathbf{x}'_i \boldsymbol{\beta} + \eta \mathbf{A}_i (\mathbf{Z} - \boldsymbol{\mu}) \}]}{1 + \exp\{ \mathbf{x}'_i \boldsymbol{\beta} + \eta \mathbf{A}_i (\mathbf{Z} - \boldsymbol{\mu}) \}'}$$

where z_i is the observed value of Z_i , and \mathbf{A}_i is the i th row of \mathbf{A} . Since the p_i are free of the normalizing function, so is the log pseudolikelihood, which is given by

$$\ell_{\text{PL}}(\boldsymbol{\theta}) = \mathbf{Z}' \{ \mathbf{X} \boldsymbol{\beta} + \eta \mathbf{A} (\mathbf{Z} - \boldsymbol{\mu}) \} - \sum_i \log[1 + \exp\{ \mathbf{x}'_i \boldsymbol{\beta} + \eta \mathbf{A}_i (\mathbf{Z} - \boldsymbol{\mu}) \}]. \quad (3)$$

Although (3) is not the true log likelihood unless $\eta = 0$, Besag (1975) showed that the MPLE converges almost surely to the maximum likelihood estimator (MLE) as the lattice size goes to ∞ (under an infill regime). For small samples the MPLE is less precise than the MLE (and the Bayes estimator), but point estimation of $\boldsymbol{\beta}$ is generally so poor for small samples that precision is unimportant. When the sample size is large enough to permit accurate estimation of $\boldsymbol{\beta}$, the MPLE is nearly as precise as the MLE (Hughes et al., 2011).

We find the MPLE $\hat{\boldsymbol{\theta}} = \arg \max \ell_{\text{PL}}(\boldsymbol{\theta})$ by using the `optim` function to minimize $-\ell_{\text{PL}}(\boldsymbol{\theta})$. Although the Nelder-Mead simplex algorithm finds the minimum quickly, we opted for the much faster BFGS method. To speed computation even further, we supply the score function

$$\nabla \ell_{\text{PL}}(\boldsymbol{\theta}) = ((\mathbf{Z} - \mathbf{p})' (\mathbf{I} - \eta \mathbf{A} \mathbf{D}) \mathbf{X}, (\mathbf{Z} - \mathbf{p})' \mathbf{A} (\mathbf{Z} - \boldsymbol{\mu}))',$$

where $\mathbf{p} = (p_1, \dots, p_n)'$ and $\mathbf{D} = \text{diag}\{\mu_i(1 - \mu_i)\}$.

Confidence intervals can be obtained using a parametric bootstrap or sandwich estimation. For the former we generate b samples from $\pi(\mathbf{Z} \mid \hat{\boldsymbol{\theta}})$ and compute the MPLE for each sample, thus obtaining the bootstrap sample $\hat{\boldsymbol{\theta}}^{(1)}, \dots, \hat{\boldsymbol{\theta}}^{(b)}$. Appropriate quantiles of the bootstrap sample are then used to construct approximate confidence intervals for the elements of $\boldsymbol{\theta}$.

The second approach for computing confidence intervals is based on (Varin et al., 2011)

$$\sqrt{n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}) \Rightarrow \mathcal{N}\{\mathbf{0}, \mathcal{I}^{-1}(\boldsymbol{\theta}) \mathcal{J}(\boldsymbol{\theta}) \mathcal{I}^{-1}(\boldsymbol{\theta})\}, \quad (4)$$

where $\mathcal{I}^{-1}(\boldsymbol{\theta}) \mathcal{J}(\boldsymbol{\theta}) \mathcal{I}^{-1}(\boldsymbol{\theta})$ is the so called Godambe sandwich matrix (Godambe, 1960). The “bread” in this sandwich is the inverse of the information matrix $\mathcal{I}(\boldsymbol{\theta}) = -\mathbb{E} \nabla^2 \ell_{\text{PL}}(\boldsymbol{\theta})$, and the “filling” is the variance of the score: $\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E} \nabla \nabla' \ell_{\text{PL}}(\boldsymbol{\theta})$. We use the observed information (computed by `optim`) in place of \mathcal{I} and estimate \mathcal{J} using a parametric bootstrap. For the bootstrap we simulate b samples $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(b)}$ from $\pi(\mathbf{Z} \mid \hat{\boldsymbol{\theta}})$ and estimate \mathcal{J} as

$$\hat{\mathcal{J}}(\hat{\boldsymbol{\theta}}) = \frac{1}{b} \sum_{k=1}^b \nabla \nabla' \ell_{\text{PL}}(\hat{\boldsymbol{\theta}} \mid \mathbf{Z}^{(k)}).$$

Because the bootstrap sample can be generated in parallel (using the `parallel` package) and little subsequent processing is required, these approaches to inference are very efficient computationally, even for large datasets. We note that sandwich estimation is over twice as fast as the full bootstrap. Moreover, asymptotic inference and bootstrap inference yield comparable results for practically all sample sizes because (4) is not, in fact, an asymptotic result. This is because the log pseudolikelihood is approximately quadratic with Hessian approximately invariant in law, which implies that the MPLE is approximately normally distributed irrespective of sample size (Geyer, 2013).

Bayesian inference for the centered autologistic model

For Bayesian inference we use the auxiliary-variable MCMC algorithm of Møller et al. (2006), which allows us to construct a proposal distribution so that the normalizing constant cancels from the Metropolis-Hastings ratio. The method requires that we can draw independent realizations from the unnormalized density for any value of $\boldsymbol{\theta}$, which we can do using our perfect sampler.

Let $h(\mathbf{Z} | \boldsymbol{\theta}) = \exp\{Q(\mathbf{Z} | \boldsymbol{\theta})\}$ denote the unnormalized density, where

$$Q(\mathbf{Z} | \boldsymbol{\theta}) = \mathbf{Z}'\mathbf{X}\boldsymbol{\beta} - \eta\mathbf{Z}'\mathbf{A}\boldsymbol{\mu} + \frac{\eta}{2}\mathbf{Z}'\mathbf{A}\mathbf{Z}$$

is the so called negpotential function; let $\mathbf{Y} \in \Omega$ denote the auxiliary variable; and let $f(\cdot)$ denote a prior distribution. Then the Metropolis-Hastings random walk acceptance probability for the algorithm of Møller et al. is given by

$$\begin{aligned} \alpha &= \frac{h(\mathbf{Y}^* | \tilde{\boldsymbol{\theta}})h(\mathbf{Z} | \boldsymbol{\theta}^*)h(\mathbf{Y} | \boldsymbol{\theta})f(\boldsymbol{\theta}^*)}{h(\mathbf{Y} | \tilde{\boldsymbol{\theta}})h(\mathbf{Z} | \boldsymbol{\theta})h(\mathbf{Y}^* | \boldsymbol{\theta}^*)f(\boldsymbol{\theta})} \frac{c(\tilde{\boldsymbol{\theta}})}{c(\boldsymbol{\theta}^*)} \frac{c(\boldsymbol{\theta})}{c(\boldsymbol{\theta}^*)} \\ &= \frac{h(\mathbf{Y}^* | \tilde{\boldsymbol{\theta}})h(\mathbf{Z} | \boldsymbol{\theta}^*)h(\mathbf{Y} | \boldsymbol{\theta})f(\boldsymbol{\theta}^*)}{h(\mathbf{Y} | \tilde{\boldsymbol{\theta}})h(\mathbf{Z} | \boldsymbol{\theta})h(\mathbf{Y}^* | \boldsymbol{\theta}^*)f(\boldsymbol{\theta})} \end{aligned}$$

where \mathbf{Y}^* is the proposed auxiliary variable, $\boldsymbol{\theta}^* = (\boldsymbol{\beta}^{*'}, \eta^{*'})'$ is the proposed $\boldsymbol{\theta}$, and $\tilde{\boldsymbol{\theta}}$ is the maximum pseudolikelihood estimate of $\boldsymbol{\theta}$. Thus $\log \alpha$ is

$$\begin{aligned} \log \alpha &= Q(\mathbf{Y}^* | \tilde{\boldsymbol{\theta}}) - Q(\mathbf{Y}^* | \boldsymbol{\theta}^*) \\ &\quad + Q(\mathbf{Z} | \boldsymbol{\theta}^*) - Q(\mathbf{Z} | \boldsymbol{\theta}) \\ &\quad + Q(\mathbf{Y} | \boldsymbol{\theta}) - Q(\mathbf{Y} | \tilde{\boldsymbol{\theta}}) \\ &\quad + \log f(\boldsymbol{\theta}^*) - \log f(\boldsymbol{\theta}) \\ &= \mathbf{Y}^{*'}\mathbf{X}(\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}^*) + \frac{\tilde{\eta} - \eta^*}{2}\mathbf{Y}^{*'}\mathbf{A}\mathbf{Y}^* + \eta^*\mathbf{Y}^{*'}\mathbf{A}\boldsymbol{\mu}^* - \tilde{\eta}\mathbf{Y}^{*'}\mathbf{A}\tilde{\boldsymbol{\mu}} \\ &\quad + \mathbf{Z}'\mathbf{X}(\boldsymbol{\beta}^* - \boldsymbol{\beta}) + \frac{\eta^* - \eta}{2}\mathbf{Z}'\mathbf{A}\mathbf{Z} + \eta\mathbf{Z}'\mathbf{A}\boldsymbol{\mu} - \eta^*\mathbf{Z}'\mathbf{A}\boldsymbol{\mu}^* \\ &\quad + \mathbf{Y}'\mathbf{X}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) + \frac{\eta - \tilde{\eta}}{2}\mathbf{Y}'\mathbf{A}\mathbf{Y} + \tilde{\eta}\mathbf{Y}'\mathbf{A}\tilde{\boldsymbol{\mu}} - \eta\mathbf{Y}'\mathbf{A}\boldsymbol{\mu} \\ &\quad + \log f(\boldsymbol{\theta}^*) - \log f(\boldsymbol{\theta}). \end{aligned} \tag{5}$$

Because the auxiliary proposals cannot be generated in parallel, this approach to Bayesian analysis is computationally expensive. Our optimized perfect sampler eases the burden somewhat, as does our Armadillo implementation of (5). We achieve an additional gain in efficiency as follows.

We use a normal random walk Metropolis-Hastings algorithm, and so our proposal for $\boldsymbol{\theta}$ is $(p + 1)$ -variate normal, i.e., $\boldsymbol{\theta}^{*(k+1)} | \boldsymbol{\theta}^{*(k)} \sim \mathcal{N}_{p+1}(\boldsymbol{\theta}^{*(k)}, \boldsymbol{\Sigma})$, for a suitably chosen covariance matrix $\boldsymbol{\Sigma}$. Taking $\boldsymbol{\Sigma} = \tau^2\mathbf{I}$ tends to result in a low acceptance rate unless τ^2 is quite small, in which case a long run is required to adequately explore the posterior. Instead, we begin by using

$$\boldsymbol{\Sigma}_0 = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{\text{GLM}} & \mathbf{0} \\ \mathbf{0} & 0.01 \end{pmatrix},$$

where $\hat{\boldsymbol{\Sigma}}_{\text{GLM}}$ is the estimated asymptotic covariance matrix for $\hat{\boldsymbol{\beta}}$ obtained from fitting the ordinary logistic model to the data. We use $\boldsymbol{\Sigma}_0$ for a training run (the number of iterations can be chosen by the user), and then we use the posterior sample covariance matrix from the training run as the proposal covariance matrix for a subsequent run. We use the latter sample to do inference. The training run usually results in a much better mixing chain, which reduces the total number of iterations. Still, rigorous Bayesian inference (following Møller et al.) is impractical for large lattices because perfect sampling does not scale well.

As for the prior $f(\boldsymbol{\theta})$, we assume that $\boldsymbol{\beta}$ and η are *a priori* independent, i.e., $f(\boldsymbol{\theta}) = f_1(\boldsymbol{\beta})f_2(\eta)$. The prior for $\boldsymbol{\beta}$ is $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The common standard deviation defaults to 1,000 but can be specified by the user via control parameter sigma. The prior for η is $\mathcal{U}(0, \eta^{\text{max}})$. The right endpoint defaults to 2 but can be chosen by the user via control parameter eta.max.

Usage examples for the centered autologistic model

Now we present some usage examples for the autologistic model. The fitting function is called autologistic and has the following signature.

```
autologistic(formula, data, A, method = c("PL", "Bayes"), model = TRUE, x = FALSE,
             y = FALSE, verbose = FALSE, control = list())
```

Arguments formula, data, model, x, and y are analogous to those that appear in the signatures of other commonly used R model fitting functions, e.g., lm and glm. A is of course the binary adjacency matrix for G. The method argument is used to select pseudolikelihood inference or Bayesian inference.

If `verbose` is equal to `TRUE`, various messages may be sent to the standard output stream. Most of those messages have to do with the default values for control parameters that were not set by the user via the control argument.

For the following example the underlying graph is the 50×50 square lattice. First the corresponding adjacency matrix is constructed. Then the vertices are assigned coordinates restricted to the unit square centered at the origin. The vertex locations are used as spatial covariates, with regression coefficients $\beta = (2, 2)'$. The resulting independence expectations $\mu = X\beta$, which range from approximately 0.1 to approximately 0.9, are level plotted in grayscale to show that the large-scale structure corresponds to a probability gradient that increases as one moves from southwest to northeast. Then, for $\eta = 0.6$, a dataset is simulated and plotted. The dataset is then fitted using the PL method, and no confidence intervals are computed. Finally, the fit is summarized.

```
R> library("ngspatial")
R> library("lattice")
R> n <- 50
R> A <- adjacency.matrix(n)
R> x <- rep(0:(n - 1) / (n - 1), times = n) - 0.5
R> y <- rep(0:(n - 1) / (n - 1), each = n) - 0.5
R> X <- cbind(x, y)
R> beta <- c(2, 2)
R> mu <- exp(X %*% beta)
R> mu <- mu / (1 + mu)
R> col1 <- "white"
R> col2 <- "black"
R> colfunc <- colorRampPalette(c(col1, col2))
R> dev.new()
R> levelplot(mu ~ x * y, aspect = "iso", col.regions = colfunc(n^2))
R> theta <- c(beta, eta = 0.6)
R> set.seed(123456)
R> Z <- rautologistic(X, A, theta)
R> dev.new()
R> levelplot(Z ~ x * y, aspect = "iso", col.regions = c("white", "black"),
+           colorkey = FALSE)
R> fit <- autologistic(Z ~ X - 1, A = A, control = list(confint = "none"))
R> summary(fit)
```

Call:

```
autologistic(formula = Z ~ X - 1, A = A, control = list(confint = "none"))
```

Control parameters:

```
confint none
```

Coefficients:

	Estimate	Lower	Upper	MCSE
Xx	1.846	NA	NA	NA
Xy	1.920	NA	NA	NA
eta	0.483	NA	NA	NA

Number of iterations: 0

The next two examples compute confidence intervals for the same dataset. The first example uses the normal approximation. The "filling" in the sandwich matrix is estimated using a parallel parametric bootstrap, where the computation is distributed across six cores on the host workstation. The second example computes bootstrap confidence intervals, where the size of the bootstrap sample is 500. The computation is once again parallelized.

```
R> set.seed(123456)
R> fit <- autologistic(Z ~ X - 1, A = A, verbose = TRUE,
+                   control = list(confint = "sandwich", nodes = 6))
R> summary(fit)
```

Control parameter 'bootit' must be a positive whole number. Setting it to the default

value of 1,000.

Control parameter 'parallel' must be a logical value. Setting it to the default value of TRUE.

Control parameter 'type' must be "SOCK", "PVM", "MPI", or "NWS". Setting it to "SOCK".

Warning: Bootstrapping may be time consuming.

Call:

```
autologistic(formula = Z ~ X - 1, A = A, verbose = TRUE,
              control = list(confint = "sandwich", nodes = 6))
```

Control parameters:

```
confint  sandwich
bootit   1000
parallel TRUE
type     SOCK
nodes    6
```

Coefficients:

	Estimate	Lower	Upper	MCSE
Xx	1.846	1.4630	2.2300	NA
Xy	1.920	1.5410	2.2990	NA
eta	0.483	0.3619	0.6042	NA

Number of iterations: 1000

```
R> set.seed(123456)
R> fit <- autologistic(Z ~ X - 1, A = A, verbose = TRUE,
+                    control = list(confint = "bootstrap", bootit = 500, nodes = 6))
R> summary(fit)
```

Control parameter 'parallel' must be a logical value. Setting it to the default value of TRUE.

Control parameter 'type' must be "SOCK", "PVM", "MPI", or "NWS". Setting it to "SOCK".

Warning: Bootstrapping may be time consuming.

Call:

```
autologistic(formula = Z ~ X - 1, A = A, verbose = TRUE,
              control = list(confint = "bootstrap", bootit = 500, nodes = 6))
```

Control parameters:

```
confint  bootstrap
bootit   500
parallel TRUE
type     SOCK
nodes    6
```

Coefficients:

	Estimate	Lower	Upper	MCSE
Xx	1.846	1.4980	2.2510	0.007607
Xy	1.920	1.5400	2.3270	0.007098
eta	0.483	0.3595	0.6063	0.003406

Number of iterations: 500

It took several minutes to run each of the two preceding examples on a Mac Pro with two quad-core 2.8 GHz Intel Xeon CPUs. Since Bayesian inference would be prohibitively expensive for a dataset so large, the next example was run on a dataset for which G was the 20×20 square lattice.

We use, and recommend, fixed-width output analysis (Flegal et al., 2008). In fixed-width analysis, one chooses a tolerance and stops sampling when all Monte Carlo standard errors fall below the tolerance. The output shown below indicates that the Monte Carlo standard errors fell below the default tolerance of 0.01 after 714,025 draws were made from the posterior, which is why the analysis was terminated before having reached the (default) maximum number of iterations (`maxit = 1e6`). We use the `batchmeans` package (Haran and Hughes, 2012) to compute Monte Carlo standard errors (denoted MCSE in the output).

In this example we use a training run of length 10,000, and the length of the subsequent inferential run will be at least 10,000.

```
R> n <- 20
R> A <- adjacency.matrix(n)
R> x <- rep(0:(n - 1) / (n - 1), times = n) - 0.5
R> y <- rep(0:(n - 1) / (n - 1), each = n) - 0.5
R> X <- cbind(x, y)
R> set.seed(123456)
R> Z <- rautologistic(X, A, theta)
R> fit <- autologistic(Z ~ X - 1, A = A, verbose = TRUE, method = "Bayes",
+               control = list(trainit = 10000, minit = 10000))
R> summary(fit)
```

Control parameter 'tol' must be a positive number. Setting it to the default value of 0.01.

Control parameter 'maxit' must be a positive whole number greater than or equal to 'minit'. Setting it to 1e+06.

Control parameter 'sigma' must be a positive number. Setting it to the default value of 1,000.

Control parameter 'eta.max' must be a positive number. Setting it to the default value of 2.

Warning: MCMC may be time consuming.

```
Progress => 5%
Progress => 10%
Progress => 15%
```

Call:

```
autologistic(formula = Z ~ X - 1, A = A, method = "Bayes", verbose = TRUE,
             control = list(trainit = 10000, minit = 10000))
```

Control parameters:

```
trainit 1e+04
tol      1e-02
minit   1e+04
maxit   1e+06
sigma   1e+03
eta.max 2e+00
```

Coefficients:

	Estimate	Lower	Upper	MCSE
Xx	2.4450	1.2120	3.667	0.009917
Xy	1.6800	0.4767	2.832	0.009800
eta	0.7363	0.4805	1.028	0.002631

Number of iterations: 714025

The object returned by function `autologistic` is a list with many useful elements. The following output resulted from applying the `names` function to the object returned in the second example above.

```
R> names(fit)

 [1] "coefficients" "fitted.values" "linear.predictors" "residuals" "convergence"
 [6] "message"      "value"          "iter"            "sample"     "mcse"
[11] "xlevels"      "call"           "terms"           "method"     "control"
[16] "model"
```

The standard `coef` and `fitted` functions can be applied to an object of type `'autologistic'`. The `residuals` method for `'autologistic'` objects can be used to extract deviance (default), Pearson, or response residuals.

```
R> library(lattice)
R> dev.new()
R> levelplot(fitted(fit) ~ x * y, aspect = "iso", col.regions = colfunc(n^2))
R> dev.new()
R> levelplot(residuals(fit) ~ x * y, aspect = "iso", col.regions = colfunc(n^2))
R> dev.new()
R> levelplot(residuals(fit, type = "pearson") ~ x * y, aspect = "iso",
+           col.regions = colfunc(n^2))
R> dev.new()
R> levelplot(residuals(fit, type = "response") ~ x * y, aspect = "iso",
+           col.regions = colfunc(n^2))
```

Other fields of interest are `value`, which contains $-\ell_{\text{PL}}(\hat{\theta})$, and `sample`, which contains the bootstrap or posterior sample.

Fitting the sparse SGLMM

The fitting function for the sparse SGLMM is called `sparse.sglm`. The function returns an object of class `"sparse.sglm"`. Auxiliary functions `residuals`, `summary`, and `vcov` accept an object of type `"sparse.sglm"` and return various kinds of residuals, summarize the fit, and return an estimated covariance matrix, respectively.

Bayesian inference for the sparse SGLMM

The second stage of the sparse SGLMM, i.e., the prior for δ_s , is

$$f(\delta_s | \tau_s) \propto \tau_s^{q/2} \exp\left(-\frac{\tau_s}{2} \delta_s' \mathbf{M}' \mathbf{Q} \mathbf{M} \delta_s\right),$$

where τ_s is a smoothing parameter. The prior for β is spherical p -variate normal with mean zero and common variance `sigma.b`, which defaults to 1,000 but can be controlled by the user via argument `hyper`. The prior for τ_s is gamma with parameters 0.5 and 2,000 (Kelsall and Wakefield, 1999). This prior is attractive because it corresponds to the prior belief that the fixed effects are sufficient to explain the data (since a large value for τ_s implies small variances for the random effects) and because it discourages artifactual spatial structure in the posterior.

When the response is normally distributed, the identity link is assumed, in which case the first stage is

$$\mu = \mathbf{X}\beta + \mathbf{M}\delta_s + \mathbf{M}\delta_h,$$

where δ_h are heterogeneity random effects. When the response is Poisson distributed, heterogeneity random effects are optional. In any case, the prior on δ_h is spherical q -variate normal with mean zero and common variance $1/\tau_h$. The prior for τ_h is gamma with parameters `a.h` and `b.h`. The default values for these parameters are 0.01 and 100, respectively, or their values can be set by the user through argument `hyper`.

If the response is Bernoulli or Poisson, β and δ_s are updated using Metropolis-Hastings random walks with normal proposals. The proposal covariance matrix for β is the estimated asymptotic covariance matrix from a `glm` fit to the data. The proposal for δ_s is spherical normal with common standard deviation `sigma.s`. If the response is Poisson distributed and heterogeneity random effects are included, those random effects are updated using a Metropolis-Hastings random walk with a spherical normal proposal. The common standard deviation is `sigma.h`. Both `sigma.s` and `sigma.h` default to 0.01, but they can be set via argument `tune`.

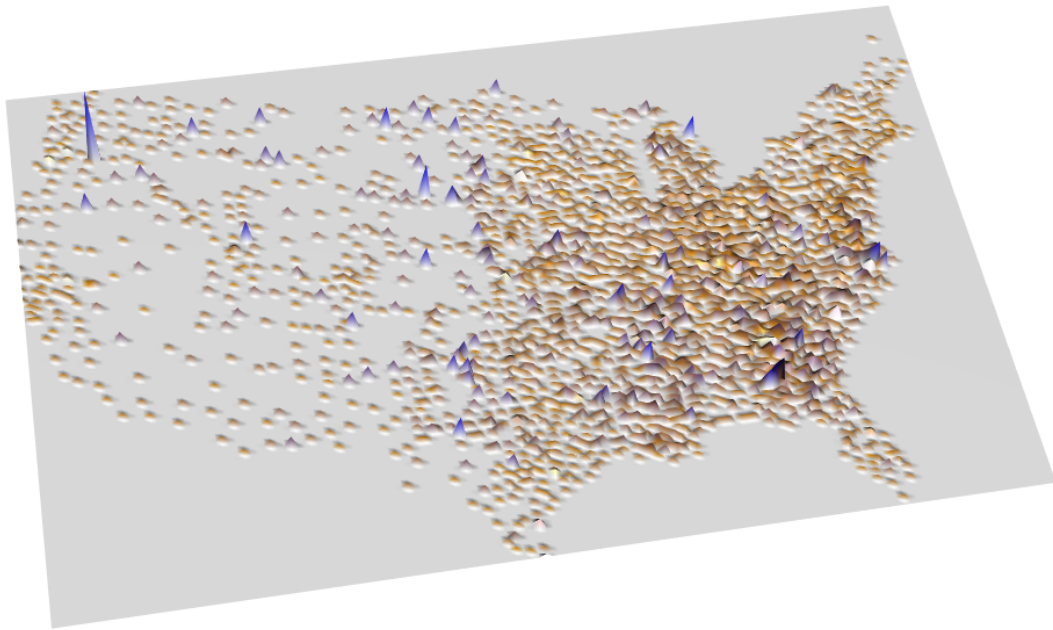


Figure 2: The empirical infant mortality rates.

The updates for τ_s and τ_t are Gibbs updates irrespective of the response distribution. If the response is normally distributed, all updates are Gibbs updates.

Usage examples for the sparse SGLMM

Function `sparse.sglm` has the following signature.

```
sparse.sglm(formula, family = gaussian, data, offset, A, attractive = 50,
  repulsive = 0, tol = 0.01, minit = 10000, maxit = 1e+06, tune = list(),
  hyper = list(), model = TRUE, x = FALSE, y = FALSE, verbose = FALSE)
```

Most of the arguments are analogous to those already described above. Arguments `attractive` and `repulsive` are used to select the number of Moran eigenvectors. Both attractive and repulsive eigenvectors are permitted, although `repulsive` defaults to 0, which corresponds to pure spatial smoothing. Function `sparse.sglm` checks the validity of `attractive` and `repulsive` by eigendecomposing the Moran operator and examining the spectrum. Execution is terminated (with an informative error message) if either value is invalid. Arguments `tune` and `hyper` can be used to supply values for tuning parameters and hyperparameters, respectively, as described above.

We will illustrate the usage of our sparse SGLMM function by analyzing the data shown in Figure 2. The plot shows infant mortality data for 3,071 US counties. Each shaded peak represents a ratio of deaths to births, i.e., an empirical infant mortality rate, for a given county. The data were obtained from the 2008 Area Resource File (ARF), a county-level database maintained by the Bureau of Health Professions, Health Resources and Services Administration, US Department of Health and Human Services. Specifically, three variables were extracted from the ARF: the three-year (2002–2004) average number of infant deaths before the first birthday, the three-year average number of live births, and the three-year average number of low birth weight infants.

To these data we fit the sparse Poisson SGLMM with

$$\log \mathbb{E}(\text{DEATHS}_i | \boldsymbol{\beta}, \boldsymbol{\delta}_s) = \log \text{BIRTHS}_i + \beta_0 + \beta_1 \text{LOW}_i + \beta_2 \text{BLACK}_i + \beta_3 \text{HISP}_i + \beta_4 \text{GINI}_i \\ + \beta_5 \text{AFF}_i + \beta_6 \text{STAB}_i + \mathbf{m}_i' \boldsymbol{\delta}_s,$$

where `DEATHS` is the number of infant deaths; `BIRTHS` is the number of live births; `LOW` is the rate of low birth weight; `BLACK` is the percentage of black residents (according to the 2000 US Census); `HISP` is the percentage of hispanic residents (2000 US Census); `GINI` is the Gini coefficient, a measure of income inequality (Gini, 1921); `AFF` is a composite score of social affluence (Yang et al., 2009); and `STAB` is residential stability, an average z-score of two variables from the 2000 US Census.

These data were analyzed previously in Hughes and Haran (2013), where it was found that the model with 50 attractive eigenvectors yields the smallest value of DIC among the sequence of models

with 25, 50, 100, and 200 attractive eigenvectors. The following code duplicates the analysis of [Hughes and Haran \(2013\)](#).

```
R> data(infant)
R> infant$low_weight <- infant$low_weight / infant$births
R> attach(infant)
R> Z <- deaths
R> X <- cbind(1, low_weight, black, hispanic, gini, affluence, stability)
R> data(A)
R> set.seed(123456)
R> fit <- sparse.sglm(Z ~ X - 1 + offset(log(births)), family = poisson, A = A,
+                    tune = list(sigma.s = 0.02), verbose = TRUE)
R> summary(fit)
```

Hyperparameter 'sigma.b' must be a positive number. Setting it to the default value of 1,000.

The Moran operator is being eigendecomposed. This operation may be time consuming.

Warning: MCMC may be time consuming.

```
Progress => 5%
.
.
.
Progress => 100%
```

Call:

```
sparse.sglm(formula = Z ~ X - 1 + offset(log(births)), family = poisson,
            A = A, tune = list(sigma.s = 0.02), verbose = TRUE)
```

Tuning parameters:

sigma.s 0.02

Hyperparameters:

sigma.b 1000

Coefficients:

	Estimate	Lower	Upper	MCSE
X	-5.435000	-5.621000	-5.250000	9.466e-04
Xlow_weight	8.821000	7.552000	10.050000	5.673e-03
Xblack	0.004184	0.002864	0.005545	6.568e-06
Xhispanic	-0.003792	-0.004870	-0.002678	6.342e-06
Xgini	-0.553200	-0.984300	-0.126000	2.070e-03
Xaffluence	-0.075600	-0.087760	-0.063740	5.148e-05
Xstability	-0.028670	-0.043350	-0.013980	6.360e-05

DIC: 10110

Number of iterations: 1000000

The analysis took just under two hours. Fitting the RHZ model to these data took approximately two weeks and required over 40 GB of secondary storage.

We also used **CARBayes** to apply the ICAR model to these data. The results for the two fits are shown together in [Table 1](#). We see that the ICAR fit suffers from spatial confounding. Specifically, some of the ICAR point estimates differ markedly from the sparse SGLMM estimates, the ICAR intervals are wider, and the Gini coefficient is not a significant predictor in the ICAR fit.

The object returned by function `sparse.sglm` is a list. The following output resulted from applying the `names` function to the object produced in the example above.

Covariate	Sparse SGLMM		ICAR	
	Estimate	95% HPD Interval	Estimate	95% HPD Interval
Intercept	-5.435	(-5.621, -5.250)	-5.558	(-5.766, -5.354)
Low Weight	8.821	(7.552, 10.050)	7.754	(6.396, 9.112)
Black	0.004	(0.003, 0.006)	0.004	(0.002, 0.006)
Hispanic	-0.004	(-0.005, -0.003)	-0.003	(-0.005, -0.002)
Gini	-0.553	(-0.984, -0.126)	-0.079	(-0.554, 0.406)
Affluence	-0.076	(-0.088, -0.064)	-0.077	(-0.091, -0.063)
Stability	-0.029	(-0.043, -0.014)	-0.042	(-0.060, -0.024)

Table 1: Results for sparse SGLMM and ICAR fits to the infant mortality data.

```
R> names(fit)
```

```
[1] "coefficients" "fitted.values" "linear.predictors" "residuals" "beta.sample"
[6] "gamma.sample" "tau.s.sample" "beta.mcse" "gamma.mcse" "tau.s.mcse"
[11] "gamma.est" "tau.s.est" "iter" "D.bar" "pD"
[16] "dic" "beta.accept" "gamma.accept" "xlevels" "call"
[21] "terms" "formula" "family" "offset" "model"
[26] "tune" "hyper"
```

The standard coef and fitted functions can be applied to an object of type ‘sparse.sglm’. The residuals method for ‘sparse.sglm’ objects can be used to extract deviance (default), Pearson, or response residuals.

Other particularly useful fields include beta.sample, gamma.sample, and tau.s.sample, which contain the posterior samples for β , δ_s , and τ_s , respectively. (The package uses γ and δ , respectively, in place of δ_s and δ_h . This paper has used δ_s and δ_h because here it was necessary to present the RHZ model, which is not discussed in the package documentation.) Monte Carlo standard errors are also returned, as are acceptance rates.

Conclusion

This article introduced version 1.0 of R package **ngspatial**, which supports two promising new models for areal data, namely, the centered autologistic model and the sparse SGLMM. The package is user friendly because its model-fitting functions and auxiliary functions were designed to behave like the analogous functions (e.g., lm, glm, summary) in the **stats** package. The package is also efficient because the code uses vector and matrix operations where possible, and because key portions of the code were written in C++.

Acknowledgments

This work was supported by a University of Minnesota Grant-in-Aid of Research, Artistry, and Scholarship.

Bibliography

- J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24(3):179–196, 1975. [p85]
- J. Besag, J. York, and A. Mollié. Bayesian image restoration, with two applications in spatial statistics. *The Annals of the Institute of Statistical Mathematics*, 43(1):1–20, 1991. [p81, 82]
- J. E. Besag. Nearest-neighbour systems and the auto-logistic model for binary data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 34(1):75–83, 1972. [p81, 82]
- R. Bivand. *spdep: Spatial Dependence: Weighting Schemes, Statistics and Models*, 2015. URL <http://CRAN.R-project.org/package=spdep>. R package version 0.5-82. [p83]
- B. Boots and M. Tiefelsdorf. Global and local spatial autocorrelation in bounded regular tessellations. *Journal of Geographical Systems*, 2(4):319, 2000. [p83]

- D. Borcard, F. Gillet, and P. Legendre. *Numerical Ecology with R*. Springer, 2011. [p84]
- P. Caragea and M. Kaiser. Autologistic models with interpretable parameters. *Journal of Agricultural, Biological, and Environmental Statistics*, 14(3):281–300, 2009. [p81, 82]
- D. Clayton, L. Bernardinelli, and C. Montomoli. Spatial correlation in ecological analysis. *International Journal of Epidemiology*, 22(6):1193–1202, 1993. [p81]
- S. Dray, P. Legendre, and P. R. Peres-Neto. Spatial modelling: A comprehensive framework for principal coordinate analysis of neighbour matrices (PCNM). *Ecological Modelling*, 196(3):483–493, 2006. [p84]
- D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL <http://www.jstatsoft.org/v40/i08>. [p84]
- J. Flegal, M. Haran, and G. Jones. Markov chain Monte Carlo: Can we trust the third significant figure? *Statistical Science*, 23(2):250–260, 2008. [p89]
- C. J. Geyer. Le Cam made simple: Asymptotics of maximum likelihood without the LLN or CLT or sample size going to infinity. In G. L. Jones and X. Shen, editors, *Advances in Modern Statistical Theory and Applications: A Festschrift in Honor of Morris L. Eaton*. Institute of Mathematical Statistics, 2013. [p85]
- C. Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121):124–126, 1921. [p91]
- V. Godambe. An optimum property of regular maximum likelihood estimation. *The Annals of Mathematical Statistics*, 31(4):1208–1211, 1960. [p85]
- D. A. Griffith. *Spatial Autocorrelation and Spatial Filtering: Gaining Understanding Through Theory and Scientific Visualization*. Springer, 2003. [p84]
- D. A. Griffith. Hidden negative spatial autocorrelation. *Journal of Geographical Systems*, 8(4):335–355, 2006. [p83]
- M. Haran and J. Hughes. *batchmeans: Consistent Batch Means Estimation of Monte Carlo Standard Errors*. Minneapolis, MN, 2012. R package version 1.0-1. [p89]
- J. Hodges and B. Reich. Adding spatially-correlated errors can mess up the fixed effect you love. *The American Statistician*, 64(4):325–334, 2010. [p81, 83]
- J. Hughes and M. Haran. Dimension reduction and alleviation of confounding for spatial generalized linear mixed models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(1): 139–159, 2013. [p81, 83, 91, 92]
- J. Hughes, M. Haran, and P. C. Caragea. Autologistic models for binary data on a lattice. *Environmetrics*, 22(7):857–871, 2011. [p82, 85]
- J. Kelsall and J. Wakefield. Discussion of ‘Bayesian models for spatially correlated disease and exposure data’, by Best et al. In J. Bernardo, J. Berger, A. Dawid, and A. Smith, editors, *Bayesian Statistics 6*. Oxford University Press, New York, 1999. [p90]
- R. Kindermann and J. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, Providence, RI, 1980. [p82]
- D. Lee. CARBayes: An R package for Bayesian spatial modeling with conditional autoregressive priors. *Journal of Statistical Software*, 55(13):1–24, 2013. URL <http://www.jstatsoft.org/v55/i13>. [p83]
- B. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80(1):221–239, 1988. [p85]
- J. Møller, A. Pettitt, K. Berthelsen, and R. Reeves. An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458, 2006. [p84, 85, 86]
- P. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23, 1950. [p83]
- I. Murray, Z. Ghahramani, and D. MacKay. MCMC for doubly-intractable distributions. *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 359–366, 2006. [p84]
- J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996. [p84]

- B. Reich, J. Hodges, and V. Zadnik. Effects of residual smoothing on the posterior of the fixed effects in disease-mapping models. *Biometrics*, 62(4):1197–1206, 2006. [p81, 83]
- H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005. [p82]
- C. Sanderson. Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, Sept. 2010. [p84]
- C. Varin, N. Reid, and D. Firth. An overview of composite likelihood methods. *Statistica Sinica*, 21(1): 5–42, 2011. [p85]
- T.-C. Yang, H.-W. Teng, and M. Haran. The impacts of social capital on infant mortality in the U.S.: A spatial investigation. *Applied Spatial Analysis and Policy*, 2(3):211–227, 2009. [p91]

John Hughes
Division of Biostatistics
School of Public Health
University of Minnesota
Minneapolis, MN 55455
USA
hughesj@umn.edu

sgof: An R Package for Multiple Testing Problems

by Irene Castro-Conde and Jacobo de Uña-Álvarez

Abstract In this paper we present a new R package called **sgof** for multiple hypothesis testing. The principal aim of this package is to implement SGoF-type multiple testing methods, known to be more powerful than the classical false discovery rate (FDR) and family-wise error rate (FWER) based methods in certain situations, particularly when the number of tests is large. This package includes Binomial and Conservative SGoF and the Bayesian and Beta-Binomial SGoF multiple testing procedures, which are adaptations of the original SGoF method to the Bayesian setting and to possibly correlated tests, respectively. The **sgof** package also implements the Benjamini-Hochberg and Benjamini-Yekutieli FDR controlling procedures. For each method the package provides (among other things) the number of rejected null hypotheses, estimation of the corresponding FDR, and the set of adjusted p values. Some automatic plots of interest are implemented too. Two real data examples are used to illustrate how **sgof** works.

Introduction

Multiple testing refers to any instance that involves the simultaneous testing of several null hypotheses, i.e.,

$$H_{01}, H_{02}, \dots, H_{0n}.$$

Nowadays, we find many statistical inference problems in areas such as genomics and proteomics which involve the simultaneous testing of thousands of null hypotheses producing as a result a number of significant p values or effects (an increase in gene expression, or RNA/protein levels). Moreover, these hypotheses may have complex and unknown dependence structures. An example from genomics is the following one involving the nulls:

$$H_{0i} : \text{Gene } i \text{ is equally expressed in groups } A \text{ and } B \ (i = 1, 2, \dots, n).$$

The goal here is to decide which H_{0i} are false, based on the p values, u_1, u_2, \dots, u_n , corresponding to a suitable test statistic (e.g., a t-test for comparison of normally distributed 'gene expression levels').

It is well known that the smaller the p value u_i , the greater the evidence against H_{0i} and, individually, H_{0i} is rejected at level α when $u_i \leq \alpha$. One of the main problems in multiple hypothesis testing is that, if one does not take the multiplicity of tests into account, then the probability that some of the true null hypotheses are rejected may be overly large. So, in the multiple testing setting, a specific procedure for deciding which null hypotheses should be rejected is needed. In this sense, the family-wise error rate (FWER) and the false discovery rate (FDR) have been proposed as suitable significance criteria to perform the multiple testing adjustment. See Benjamini and Hochberg (1995), Nichols and Hayasaka (2003) or Dudoit and van der Laan (2007) for more information. But the FDR and FWER based methods have the drawback of a rapidly decreasing power as the number of tests grows, being unable to detect even one effect in particular situations such as when there is a small to moderate proportion of weak effects.

In this paper we introduce the **sgof** package which implements, for the first time in R, SGoF-type methods (Carvajal-Rodríguez et al., 2009; de Uña-Álvarez, 2011), which have been proved to be more powerful than FDR and FWER based methods in certain situations, particularly when the number of tests is large (Castro-Conde and de Uña-Álvarez, in press). BH (Benjamini and Hochberg, 1995) and BY (Benjamini and Yekutieli, 2001) methods are included in the package for completeness. Users can easily obtain from this package a complete list of results of interest in the multiple testing context. The original SGoF procedure (Carvajal-Rodríguez et al., 2009) is also implemented in the GNU software SGoF+ (Carvajal-Rodríguez and de Uña-Álvarez, 2011), see <http://webs.uvigo.es/acraaj/SGoF.htm>, while a MATLAB version was also developed (Thompson, 2010). However, none of these tools work within R, nor do they include the several existing corrections of SGoF for dependent tests. These limitations are overcome by package **sgof**.

Recent contributions in which the SGoF method has been found to be a very useful tool include protein evolution (Ladner et al., 2012) and neuroimaging (Thompson et al., 2014).

Existing software

The Bioconductor software (Gentleman et al., 2004) provides tools for the analysis and comprehension of high-throughput genomics data. Bioconductor uses the R statistical programming language and is open source and open development. It has two releases each year, nearly thousand software packages, and an active user community. Some of the tools of Bioconductor related to multiple testing methods are the following.

1. The **qvalue** package (Dabney and Storey, 2014) takes the list of p values and estimates their q -values. The q -value of a test measures the proportion of false positives incurred when that particular test is called significant. Various plots are automatically generated, allowing one to make sensible significance cutoffs.
2. The **HybridMTest** package (Pounds and Fofana, 2011) performs hybrid multiple testing that incorporates method selection and assumption evaluations into the analysis using empirical Bayes probability estimates obtained by Grenander density estimation.
3. The **multtest** package (Pollard et al., 2005) performs non-parametric bootstrap and permutation resampling-based multiple testing procedures (including empirical Bayes methods) for controlling the FWER, generalized FWER, tail probability of the proportion of false positives, and FDR. Results are reported in terms of adjusted p values, confidence regions and test statistic cutoffs. The procedures are directly applicable to identify differentially expressed genes in DNA microarray experiments.

Other R packages for multiple testing problems include the following.

1. The **mutoss** package (MuToss Coding Team et al., 2014) is designed to the application and comparison of multiple hypotheses testing procedures like the LSL method presented in Hochberg and Benjamini (1990) or the Storey et al. (2004) adaptive step-up procedure.
2. The **multcomp** package (Hothorn et al., 2008) performs simultaneous tests and confidence intervals for general linear hypotheses in parametric models, including linear, generalized linear, linear mixed effects and survival models.
3. The stats package includes the function `p.adjust` which, given a set of p values, returns adjusted p values using one of several methods like "holm" (Holm, 1979), "hochberg" (Hochberg, 1988), "hommel" (Hommel, 1988) and "BH" (Benjamini and Hochberg, 1995).

The rest of the paper is organized as follows. First we introduce the methodological background for SGoF- and FDR-type methods. Then the **sgof** package is described and its usage is illustrated through the analysis of two real data sets. Finally, the last section contains the main conclusions of this work.

Methodology

SGoF multiple testing procedure

Carvajal-Rodríguez et al. (2009) proposed a new multiple comparisons adjustment, called SGoF (from sequential goodness-of-fit) which is based on the idea of comparing the number of p values falling below an initial significance threshold γ (typically $\gamma = 0.05, 0.01, 0.001$) to the expected amount under the complete null hypothesis that all the n nulls are true (i.e., no effects), which is $n\gamma$.

In order to formalize things, let F and F_n be the underlying distribution function of the p values and their empirical distribution, respectively. SGoF multitest performs a standard one-sided binomial test (we will refer to this method as Binomial SGoF) for $H_0 : F(\gamma) = \gamma$; therefore, H_0 is rejected at level α if and only if $nF_n(\gamma) \geq b_{n,\alpha}(\gamma)$, where

$$b_{n,\alpha}(\gamma) = \inf \{b \in \{0, \dots, n\} : P(\text{Bin}(n, \gamma) \geq b) \leq \alpha\} \quad (1)$$

is the $(1 - \alpha)$ -quantile of the $\text{Bin}(n, \gamma)$ distribution. This relates to the notion of higher criticism introduced by Tukey (1976). When H_0 is rejected, the null hypotheses corresponding to the $N_{n,\alpha}(\gamma) = nF_n(\gamma) - b_{n,\alpha}(\gamma) + 1$ smallest p values are declared as false by Binomial SGoF, which is just the excess of significant cases in the binomial test. Note that, when n is large, $N_{n,\alpha}(\gamma)$ approximates $n[F_n(\gamma) - \gamma] - n\sqrt{\text{Var}^{(0)}(F_n(\gamma))}z_\alpha + 1$, where $\text{Var}^{(0)}(F_n(\gamma)) = \gamma(1 - \gamma)/n$ and z_α is the $(1 - \alpha)$ -quantile of the standard normal.

A slightly different version of the SGoF procedure is obtained when declaring as true effects the $N_{n,\alpha}^{(1)}(\gamma) = n[F_n(\gamma) - \gamma] - n\sqrt{\text{Var}^{(1)}(F_n(\gamma))}z_\alpha + 1$ smallest p values, where the variance of the

proportion of p values below gamma ($Var^{(1)}(F_n(\gamma)) = F_n(\gamma)(1 - F_n(\gamma))/n$) is estimated without assuming that all the null hypotheses are true; this typically results in a more conservative decision (thus the method's name). Since Conservative SGoF is based on the asymptotic binomial-normal approximation, it should not be used when the number of tests is small. When the number of tests is large, Conservative SGoF will often approximate Binomial SGoF since the variance term has a smaller order of magnitude compared to $n[F_n(\gamma) - \gamma]$.

The main properties of SGoF-type procedures were analyzed in detail by de Uña-Álvarez (2011, 2012). In particular, it was shown that SGoF gives flexibility to the FDR by controlling it at level α only under the complete null (i.e. weak control of FDR), which results in an increasing power compared to classical FDR controlling methods. It was also shown that the power of SGoF procedures increases with the number of tests, and that α is a bound for the undesirable event that the number of false positives is greater than the number of false negatives among the p values smaller than γ .

Bayesian SGoF procedure

The SGoF Bayesian procedure is an adaptation of the original SGoF to the Bayesian paradigm (Castro-Conde and de Uña-Álvarez, 2013). In this context, it is assumed that the probability $\theta = P(u_i \leq \gamma) = F(\gamma)$ follows a prior density $\pi(\theta)$ supported on the unit interval. The relevant 'sample information' is given by $\vec{x} = (I_{\{u_1 \leq \gamma\}}, \dots, I_{\{u_n \leq \gamma\}})$.

The Bayesian SGoF procedure consists of two main steps. In the first step Bayesian SGoF decides if the complete null hypothesis is true or false, by using a pre-test rule which works as follows. First, the usual default prior probabilities for H_0 and H_1 ($P_0 = P_1 = 1/2$) are taken and a beta distribution, with location γ and dispersion parameter ρ , is assumed as prior distribution of θ under the alternative. Then, using this a priori information, the posterior probability that H_0 is true $P(H_0 | \vec{x})$ is computed. Since $P(H_0 | \vec{x})$ heavily depends on the dispersion of the beta prior, lower bounds $\underline{P}(H_0 | \vec{x}) = \inf_{\rho} P(H_0 | \vec{x})$ are used in practice. $\underline{P}(H_0 | \vec{x})$ has been proposed as the suitable way of looking for evidence against a point null hypothesis in the Bayesian setting (Berger and Delampady, 1987; Berger and Sellke, 1987). Let s be the number of p values below γ , and let $s_{\alpha} - 1$ be the first value of s when going from n to 0 for which $\underline{P}(H_0 | \vec{x}) \geq \alpha$ (note that \vec{x} essentially reduces to s). The complete null hypothesis is rejected when $s \geq s_{\alpha}$. Of course, s_{α} may be done dependent on others choices for P_0 by including them in the computation of $\underline{P}(H_0 | \vec{x})$ (see Section [Small number of tests: Needleman data](#) for illustration).

The second step in Bayesian SGoF is to compute the number of rejected nulls. Proceeding analogously to the frequentist SGoF, a one-sided $100(1 - \alpha)\%$ credible interval for θ is constructed. Let $l_{\alpha}(\pi, \vec{x})$ be the α -quantile of the posterior density $\pi(\theta | \vec{x})$ computed from the non-informative prior $\pi(\theta)$ (i.e., the uniform density). Then, the Bayesian SGoF method declares as non-true the null hypotheses with the smallest $N_{n,\alpha}^b(\gamma)$ p values (which represents the 'excess of significant cases'), where $N_{n,\alpha}^b(\gamma) = \max(n(l_{\alpha}(\pi, \vec{x}) - \gamma), 0)$. When the pre-test does not reject the complete null ($s < s_{\alpha}$), this is automatically set to zero.

Compared to frequentist versions of SGoF (Binomial SGoF, Conservative SGoF), Bayesian SGoF may result in a more conservative approach, particularly when the number of tests is low to moderate. This is so because of the Bayesian perspective for testing for point nulls, on which the pre-test rule is based. That is, Bayesian SGoF will accept the absence of features in situations when classical SGoF detects a signal. Another feature of Bayesian SGoF is the interpretation of the results. It should be taken into account that Bayesian SGoF controls for the probability of type I errors conditionally on the given set of p values and, hence, it refers to all the situations with the same amount of evidence as the data at hand. This departs from frequentist methods which aim to control error rates when averaging the results among all the possible samples. On the other hand, as n grows, the sampling information becomes more relevant and, accordingly, the prior density has a vanishing effect; from this, it is not surprising that Bayesian SGoF and frequentist SGoF may report similar results in large number of tests settings.

Beta-Binomial SGoF procedure

It has been quoted that the SGoF multiple testing procedure is very sensitive to correlation among the tests, in the sense that it may become too liberal when the p values are dependent (Carvajal-Rodríguez et al., 2009). A correction of SGoF for serially dependent tests was proposed in de Uña-Álvarez (2012). Since the correction is based on a beta-binomial model (which is an extension of the binomial model allowing for positive correlation), it is termed Beta-Binomial SGoF or BB-SGoF. A quick description of the main ideas behind BB-SGoF is the following.

Given the initial significance threshold γ , BB-SGoF transforms the original p values u_1, \dots, u_n into n realizations of a Bernoulli variable: $X_i = I_{\{u_i \leq \gamma\}}, i = 1, \dots, n$; and assumes that there are

k independent blocks of p values. Then, the number of successes ($X_i = 1$) s_j within each block j , $j = 1, \dots, k$, is computed, where s_j is assumed to be a realization of a beta-binomial variable with parameters (n_j, θ, ρ) . Here n_j is the size of the block j , $\theta = F(\gamma)$, and ρ is the correlation between two different indicators X_i and X_j inside the same block (the within-block correlation). Note that, if $\rho = 0$, then we come back to the binomial model and therefore to the original SGoF method for independent tests.

Analogously to original SGoF, BB-SGoF proceeds by computing a one-sided confidence interval for the difference between the observed and expected amounts of p values below γ . For this, however, estimates based on a beta-binomial (rather than binomial) likelihood and their standard errors are obtained; the bounds are easily computed from the asymptotic normal theory for maximum-likelihood estimation. Of course, due to the allowed correlation, there is a variance increase which results in a smaller amount of rejected nulls, $N_\alpha^{BB}(\gamma; k)$ let's say. This is more evident when the number of existing blocks is small (stronger dependence structure). Otherwise, BB-SGoF shares the main properties of the SGoF procedure regarding weak control of FWER and relatively large power, which increases with n . An extensive simulation study on the method's performance was provided in [Castro-Conde and de Uña-Álvarez \(in press\)](#).

A practical issue in the application of BB-SGoF is the choice of the number and the size of the blocks (otherwise these blocks are assumed to be located following the given sequence of p values, which therefore should not be sorted before their analysis). As a compromise, BB-SGoF takes blocks of the same size. Regarding the number of blocks k , a data-driven solution is the automatic choice $k_N = \arg \min_k N_\alpha^{BB}(\gamma; k)$, corresponding to the most conservative decision of declaring the smallest number of effects along a grid of k -values. This of course may (and will) entail some extra loss of power. In order to mitigate this, a preliminary test of no correlation is recommended; in the setting of the beta-binomial model, such a test was suggested by [Tarone \(1979\)](#). When the null hypothesis of zero correlation is accepted, one goes back to the application of Binomial or Conservative SGoF methods for independent tests.

FDR-controlling step-up procedures

Unlike SGoF-type procedures, FDR based methods aim to control the expected proportion of false discoveries at a given level α . The Benjamini-Hochberg (BH; [Benjamini and Hochberg, 1995](#)) step-up procedure achieves this by proceeding as follows:

1. For a given α , let j be the largest i for which $u_{(i)} \leq \frac{i}{n}\alpha$, where $u_{(1)} \leq u_{(2)} \leq \dots \leq u_{(n)}$ are the ordered p values.
2. Then reject (i.e., declare positive discoveries) all $H_{0(i)}$ for $i = 1, 2, \dots, j$, where $H_{0(i)}$ is the null hypothesis attached to $u_{(i)}$.

The BH procedure controls the FDR at level α when the n tests are independent or in the case of positive regression dependence. [Benjamini and Yekutieli \(2001\)](#) introduced an alternative procedure (termed BY in this paper) which ensures FDR control under arbitrary forms of dependence. BY proceeds similarly to BH but replaces $u_{(i)} \leq \frac{i}{n}\alpha$ by $u_{(i)} \leq \frac{i}{n \sum_{l=1}^i 1/l} \alpha$ in Step 1 above. Obviously, this results in a more conservative decision on the number of non-true nulls.

FDR based methods are often used nowadays to take the multiplicity of tests into account. However, as mentioned, they may exhibit a poor power in particular scenarios, namely, those with a large number of tests and a small to moderate proportion of 'weak effects' (true alternatives close to the corresponding nulls). In such settings, application of alternative methods like SGoF-type procedures is recommended.

The q-value procedure

A method closely related to BH is the q-value approach ([Storey, 2003](#)). The q-value of an individual test is the expected proportion of false positives incurred when calling that test significant. Formally, define the positive false discovery rate (pFDR) as follows:

$$pFDR = E \left(\frac{V}{R} \mid R > 0 \right),$$

where V and R stand for the number of type I errors and the number of rejections, respectively. For a nested set of rejection regions $\{\Gamma_\alpha\}_{\alpha=0}^1$ and for an observed statistic $T = t$, the q-value of t is defined to be:

$$q(t) = \inf_{\{\Gamma_\alpha: t \in \Gamma_\alpha\}} \{pFDR(\Gamma_\alpha)\}$$

Therefore, the q-value is the minimum possible pFDR when rejecting a statistic with value t for the set of nested significance regions.

The q-value procedure rejects all the null hypotheses with a q-value below the nominal level α , attaining a $FDR \leq \alpha$ under weak dependence (Storey and Tibshirani, 2003). According to this definition, the q-value method may report power gains compared to the standard BH procedure.

Adjusted p values

A very important concept in the multiple testing context is that of adjusted p values. The adjusted p value \tilde{u}_i , for null hypothesis H_{0i} with p value u_i , is the smallest level of the multiple testing procedure at which H_{0i} is still rejected. Adjusted p values for the Binomial SGoF method were introduced by Castro-Conde and de Uña-Álvarez (2015) by linking the significance threshold γ and the level at which the binomial test is performed. This gives the following definition for \tilde{u}_i :

$$\tilde{u}_i \equiv \inf\{\alpha \in [0, 1] : nF_n(u_i) \leq N_{n,\alpha}(\alpha)\}, \quad \text{if } \{\alpha \in [0, 1] : nF_n(u_i) \leq N_{n,\alpha}(\alpha)\} \neq \emptyset.$$

$$\tilde{u}_i \equiv 1, \quad \text{otherwise, } \quad i = 1, \dots, n.$$

Adjusted p values for the other SGoF-type methods may be defined in the same way. Note that, however, this definition entails the searching for an infimum, which may be very computationally intensive, particularly when n is large. In order to speed up the procedure, we approximate the infimum by a minimum over the set of original p values: $\alpha \in \{u_1, \dots, u_n\}$. Interestingly, Castro-Conde and de Uña-Álvarez (2015) proved that this simplification does not induce any real change in the definition of the adjusted p values for Binomial SGoF (since the infimum is attained on the set of p values). For other methods there is no such result but, clearly, a sufficiently good approximation is expected as the number of tests grow. Regarding the interpretation of the \tilde{u}_i 's note that, when an adjusted p value is smaller than or equal to u , then it is known that there exists $\eta \leq u$ such that the corresponding SGoF-type method based on $\gamma = \alpha = \eta$ rejects the null; this does not imply by force that the null will be also rejected at level u since the number of rejections of SGoF-type methods is roughly a concave function of the significance threshold, increasing up to a maximum and then decreasing.

On the other hand, the adjusted p values of the BH and BY methods are defined, respectively, as follows (Dudoit and van der Laan, 2007):

$$\tilde{u}_i^{BH} \equiv \min_{h \in [i, \dots, n]} \{\min\{\frac{n}{h}u_i, 1\}\} \quad i = 1, \dots, n.$$

$$\tilde{u}_i^{BY} \equiv \min_{h \in [i, \dots, n]} \{\min\{(\sum_{i=1}^n 1/i) \frac{n}{h}u_i, 1\}\} \quad i = 1, \dots, n.$$

In this case, the adjusted p value gives information about the minimum possible FDR when declaring it as a true effect.

Package **sgof** in practice

As mentioned, the **sgof** package implements different procedures for solving multiple testing problems. This section illustrates the usage of **sgof** by describing its main features and by analyzing two real data sets. The first data set refers to a situation in which the number of tests (n) is small; the tests correspond to a sequence of 11 p values coming from a study of the neuropsychologic effects of unidentified childhood exposure to lead performances between two groups of children. The second example of application is related to a large number of test settings where more than 3,000 tests are performed, corresponding to the comparison of mean gene expression levels in two groups of patients. This second data set is included in the **sgof** package as Hedenfalk. The package **sgof** implements for the first time the four SGoF-type methods which have been reviewed in the previous section.

The **sgof** package includes six functions: Binomial.SGoF, SGoF, Bayesian.SGoF, BBSGoF, BH and BY. All of the six functions estimate the FDR by the simple method proposed by Dalmaso et al. (2005) by taking $n = 1$ in their formula. The structure and performance of the six functions are summarized below. More information is available in the package documentation.

Table 1 shows a list of the arguments in the six functions. It should be noted that only the argument u (the vector of p values) is a required argument since the other ones have a default value. This is the reason why all the functions check the arguments. For example, if the user forgets to write the argument u in the function `SGoF()`, the following message will be returned:

```
> SGoF(alpha = 0.05, gamma = 0.05)
Error in SGoF(alpha = 0.05, gamma = 0.05) : data argument is required
```

BH() and BY() arguments	
u	The (non-empty) numeric vector of p values.
alpha	Numerical value. The significance level of the metatest. Default is $\alpha = 0.05$.
Binomial.SGoF(), SGoF(), Bayesian.SGoF() and BBSGoF() arguments	
u	The (non-empty) numeric vector of p values.
alpha	Numerical value. The significance level of the metatest. Default is $\alpha = 0.05$.
gamma	Numerical value. The p value threshold, so that the SGoF-type method looks for significance in the amount of p values below γ . Default is $\gamma = 0.05$.
Bayesian.SGoF() arguments	
P0	Numerical value. The a priori probability of the null hypothesis. Default is $P0 = 0.5$.
a0	Numerical value. The first parameter of the a priori beta distribution. Default is $a0 = 1$.
b0	Numerical value. The second parameter of the a priori beta distribution. Default is $b0 = 1$.
BBSGoF() arguments	
kmin	Numerical value. The smallest allowed number of blocks of correlated tests. Default is $kmin = 2$.
kmax	Numerical value. The largest allowed number of blocks of correlated tests. Default is $kmax = \min(\text{length}(u)/10, 100)$.
tol	Numerical value. The tolerance in model fitting. Default is $tol = 10$. It allows for a stronger (small tol) or weaker (large tol) criterion when removing poor fits of the beta-binomial model. When the variance of the estimated beta-binomial parameters for a given k is larger than tol times the median variance along $k = kmin, \dots, kmax$, the particular value of k is discarded.
adjusted.pvalues	Logical. Default is FALSE. If TRUE, the adjusted p values are computed.
blocks	Numerical value. The number of existing blocks in order to compute the adjusted p values.

Table 1: Arguments of the six functions of the `sgof` package.

Moreover, in the event the user chooses the option `adjusted.pvalues = TRUE` in the function `BBSGoF()` and forgets to write the argument `blocks`, then this function will return the following message:

```
> BBSGoF(u, adjusted.pvalues = TRUE)
Error in BBSGoF(u, adjusted.pvalues = TRUE) :
blocks argument is required to compute the Adjusted p-values
```

Note also that `kmax` should be larger than `kmin` and smaller than the number of tests n (if the number of blocks is n then one is indeed assuming independence and we should rather use `SGoF()` instead of `BBSGoF()`), otherwise `BBSGoF()` will return the following messages:

```
> BBSGoF(u, kmin = 5, kmax = 3)
Error in BBSGoF(u, kmin = 5, kmax = 3) : kmax should be larger than kmin

> BBSGoF(u, kmax = length(u))
Error in BBSGoF(u, kmax = length(u)) : kmax should be lower than n
```

Finally, note that `BBSGoF()` usually returns a warning message indicating which blocks k are removed because they provided negative or atypical variance estimates. The set of removed blocks depends on the parameter `tol` which allows for a stronger or weaker criterion when removing poor fits of the beta-binomial model (see Section [Package `sgof` in practice](#) for an example).

On the other hand, Table 2 shows a summary of the results given by each of the functions. It can be seen that the number of rejections and the estimation of the FDR are a common returned value whereas the adjusted p values are computed by every function except by `Bayesian.SGoF()`. Moreover, the `Bayesian.SGoF()` function also computes the posterior probability that the complete null hypothesis is true, based on the default *a priori* probabilities $P0 = P1 = 1/2$ and the non-informative prior $\pi(\theta) = 1$ (unless indicated otherwise), as well as the amount of p values falling below γ (`s`) and the critical point at level α for the Bayesian pre-test for the complete null (`s.alpha`). Finally, the `BBSGoF()` function also computes some parameters of interest like (among others) a vector with the number of effects declared by `BBSGoF()` for each value of k (`effects`), the automatic number of blocks (`automatic.blocks`), a vector with the values of k for which the model fitted well (`n.blocks`), a vector with the estimated within-block correlation (`cor`), a vector with the p values of Tarone's test for no correlation (`Tarone.pvalues`), and the estimated parameters of the Beta(a, b) and Betabinomial(p, ρ) models for the automatic k .

Finally, the `sgof` package implements three different methods for the returned objects of classes `'Binomial.SGoF'`, `'SGoF'`, `'BBSGoF'`, `'BH'` and `'BY'` classes. The `print` method which prints the correspond-

	Binomial.SGoF(), SGoF(), Bayesian.SGoF(), BBSGoF(), BH() and BY()
Rejections	The number of declared effects.
FDR	The estimated false discovery rate.
Adjusted.pvalues	The adjusted p values*.
	BBSGoF()
effects	A vector with the number of effects declared by BBSGoF() for each value of k .
SGoF	The number of effects declared by SGoF().
automatic.blocks	The automatic number of blocks.
deleted.blocks	A vector with the values of k for which the model gave a poor fit.
n.blocks	A vector with the values of k for which the model fitted well.
p	The average ratio of p values below gamma.
cor	A vector with the estimated within-block correlation.
Tarone.pvalues	A vector with the p values of Tarone's test for no correlation.
Tarone.pvalue.auto	The p values of Tarone's test for the automatic k .
beta.parameters	The estimated parameters of the Beta(a , b) model for the automatic k .
betabinomial.parameters	The estimated parameters of the Betabinomial(p , ρ) model for the automatic k .
sd.betabinomial.parameters	The standard deviation of the estimated parameters of the Betabinomial(p , ρ) model for the automatic k .
	Bayesian.SGoF()
Posterior	The posterior probability that the complete null hypothesis is true considering the prior information a_0 , b_0 and P_0 .
s	The amount of p values falling below gamma.
s.alpha	Critical point at level alpha of the Bayesian pre-test for the complete null depending on P_0 .

Table 2: Summary of the results reported by the six functions of the **sgof** package.

* Bayesian.SGoF() does not compute the adjusted p values.

ing object in a nice way, the summary method which prints a summary of the main results reported, and the plot method which provides a graphical representation of the adjusted p values versus the original ones; and, in the case of BBSGoF(), four more plots of interest: the fitted beta density, the Tarone's p values, the number of effects, and the within-block correlation for each particular number of blocks in the grid (except the deleted ones). As an exception, the 'Bayesian.SGoF' class does not have a plot method as the adjusted p values given by the Bayesian SGoF procedure are not computed.

Small number of tests: Needleman data

Needleman et al. (1979) compared various psychological and classroom performances between two groups of children in order to study the neuropsychologic effects of unidentified childhood exposure to lead. Needleman's study was attacked because it presented three families of endpoints but carried out separate multiplicity adjustments within each family. For illustration of **sgof**, we will focus on the family of endpoints corresponding to the teacher's behavioral ratings. Table 3 shows the original p values (saved in the vector u) as well as the adjusted p values reported by the BH() and Binomial.SGoF() functions, computed using the following code:

```
> u <- c(0.003, 0.003, 0.003, 0.01, 0.01, 0.04, 0.05, 0.05, 0.05, 0.08, 0.14)
> BH(u)$Adjusted.pvalues
> Binomial.SGoF(u)$Adjusted.pvalues
```

Note that tied p values are present in this data set; in particular, it is clear that one would reject 3, 5, 6, 9, 10 or 11 nulls, depending on the level. On the other hand, there are 9 p values below 0.05, which is greater than the expected amount under the complete null (0.55).

We will use Needleman's p values (that we saved in the vector u) to illustrate the performance of the BH(), Binomial.SGoF() and Bayesian.SGoF() functions, using default argument values. SGoF() and BBSGoF() are not applied in this case because these are asymptotic methods and here the sample size is small ($n = 11$).

The first step to analyze Needleman data is to load the **sgof** package by using the code line: library(sgoF). We start then by applying the BH() function:

```
> m1 <- BH(u)
> summary(m1)
```

	<i>p</i> values	Adjusted <i>p</i> values	
		BH	Binomial.SGoF
Distractible	0.003	0.011	0.010
Does not follows sequence of directions	0.003	0.011	0.010
Low overall functioning	0.003	0.011	0.010
Impulsive	0.010	0.022	0.050
Daydreamer	0.010	0.022	0.050
Easily frustrated	0.040	0.061	0.050
Not persistent	0.050	0.061	1.000
Dependent	0.050	0.061	1.000
Does not follow simple directions	0.050	0.061	1.000
Hyperactive	0.080	0.088	1.000
Disorganized	0.140	0.140	1.000

Table 3: Needleman data.

```
Call:
BH(u = u)
```

```
Parameters:
alpha= 0.05
```

```
$Rejections
[1] 5
```

```
$FDR
[1] 9e-04
```

```
$Adjusted.pvalues
>alpha <=alpha
  6      5
```

The output of the summary shows that the BH procedure with 5% FDR control is rejecting 5 null hypotheses (corresponding to the five smallest *p* values) with a estimated FDR of 0.09%. Besides, the summary indicates that there are five adjusted *p* values falling below alpha, which is by force the case. If we apply the BY procedure to this set of *p* values we obtain a number of rejections (3) smaller than that given by BH, something expected since BY takes the dependence into account.

```
> BY(u)$Rejections
```

```
[1] 3
```

Now, we illustrate the usage of the Binomial.SGoF() function:

```
> m2 <- Binomial.SGoF(u)
> summary(m2)
```

```
Call:
Binomial.SGoF(u = u)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
```

```
$Rejections
[1] 6
```

```
$FDR
[1] 0.0031
```

```
$Adjusted.pvalues
>gamma <=gamma
  5      6
```

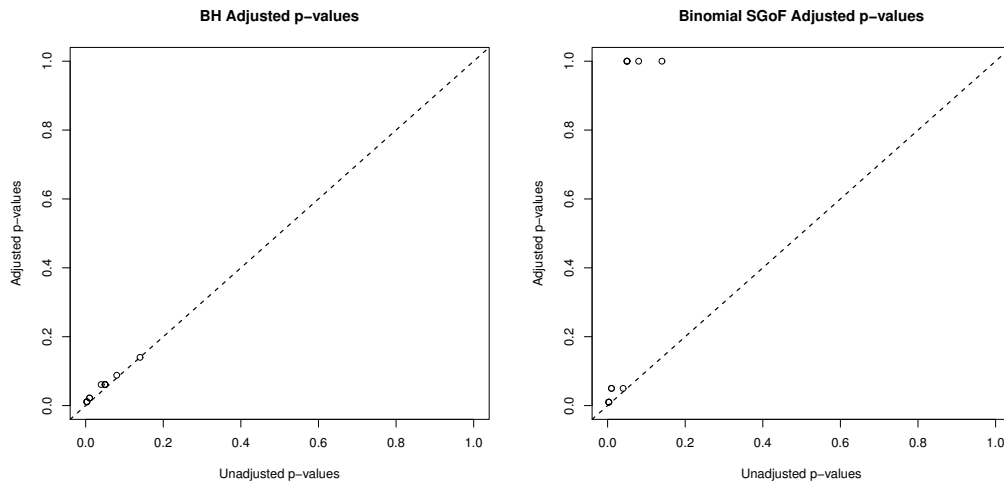



Figure 1: Needleman data. Adjusted p values reported by BH (left) and Binomial SGoF (right) methods versus the original ones.

In this case, the summary indicates that the default Binomial SGoF procedure ($\alpha = \gamma = 0.05$) declares six effects (estimated FDR of 0.31%), which represents one rejection more than the BH method. We should also point out that, in this example, the number of adjusted p values below gamma is equal to the number of rejections, which will not be the case in general (recall that the number of rejections of SGoF is an increasing-decreasing function of γ). When an adjusted p value is smaller than the threshold γ , what one actually knows is that there exists some $\gamma' \leq \gamma$ such that the corresponding null H_{0i} is rejected by SGoF at level $\alpha' = \gamma'$.

Figure 1 reports the graphical displays from the plot method applied to the m1 and m2 objects (plot(m1), plot(m2)), where the adjusted p values versus the original ones are depicted.

In the multiple testing setting, these plots are often used to inspect the relative size and distribution of the adjusted p values. Of course, all the points in these plots fall above the diagonal. The adjusted p values are also given in Table 3, where we see that the first six adjusted p values of Binomial SGoF method are smaller than those pertaining to BH, suggesting that SGoF entails an increase in power.

Results of Bayesian.SGoF() are as follows:

```
> m3 <- Bayesian.SGoF(u)
> summary(m3)
```

```
Call:
Bayesian.SGoF(u = u)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
P0= 0.5
a0= 1
b0= 1
```

```
$Rejections
[1] 6
```

```
$FDR
[1] 0.0031
```

```
$Posterior
[1] 0
```

```
$s
[1] 9
```

```
$s.alpha
```



```
[1] 5
```

By using the Bayesian.SGoF() function one obtains the same number of declared effects and estimated FDR as those reported by the Binomial SGoF procedure. Besides, the summary of the 'Bayesian.SGoF' object shows that, while there are nine original p values falling below γ , the critical point at level α for the Bayesian pre-test is five, which is lower than s as expected (if $s \cdot \alpha > s$ then Bayesian SGoF would have accepted the complete null). Besides, the posterior probability that the complete null is true is zero. By choosing the default values of Bayesian.SGoF() one considers as non-informative $\pi(\theta)$ the uniform density in the $[0,1]$ interval. When there is *a priori* information on this distribution then the arguments a_0 and b_0 may be used to include such information. Below we provide the results when choosing $a_0 = 2$ and $b_0 = 8$, which corresponds to a Beta(2, 8) distribution with mean 0.2 (the mean of the default distribution is 0.5). It is seen that this leads, as expected, to a fewer number of rejections (3). Note that $s \cdot \alpha$ is not depending on a_0 and b_0 .

```
> m32 <- Bayesian.SGoF(u, a0 = 2, b0 = 8)
> summary(m32)
```

```
Call:
```

```
Bayesian.SGoF(u = u, a0 = 2, b0 = 8)
```

```
Parameters:
```

```
alpha= 0.05
```

```
gamma= 0.05
```

```
P0= 0.5
```

```
a0= 2
```

```
b0= 8
```

```
$Rejections
```

```
[1] 3
```

```
$FDR
```

```
[1] 5e-04
```

```
$Posterior
```

```
[1] 0
```

```
$s
```

```
[1] 9
```

```
$s.alpha
```

```
[1] 5
```

Now, by choosing $P_0 = 0.2$ to represent a small *a priori* probability that the complete null is true, one obtains the same number of rejections but the lower bound of the Bayesian pre-test changes (which depends on P_0 but not on a_0 or b_0), being $s \cdot \alpha = 3$. That is, if the number of existing p values below γ were 4 (rather than 9), the complete null would be rejected with $P_0 = 0.2$ but not with the default option $P_0 = 0.5$. This means that, by choosing a lower a priori probability, P_0 , Bayesian SGoF is more likely to reject the complete null hypothesis.

```
> m33 <- Bayesian.SGoF(u, a0 = 2, b0 = 8, P0 = 0.2)
> summary(m33)
```

```
...
```

```
$Rejections
```

```
[1] 3
```

```
$FDR
```

```
[1] 5e-04
```

```
$Posterior
```

```
[1] 0
```

```
$s
```

```
[1] 9
```

```
$s.alpha
[1] 3
```

In order to illustrate how some of these results change when changing the value for the argument `alpha`, we apply the `Binomial.SGoF()`, `Bayesian.SGoF()`, `BH()` and `BY()` functions to the Needleman data with `alpha = 0.01`. While the number of rejections reported by the Binomial SGoF procedure remains the same, Bayesian SGoF becomes more conservative declaring one less effect. Stronger consequences are found for BH and BY procedures, which are unable to find any effect with such a restrictive FDR level.

```
> Binomial.SGoF(u, alpha = 0.01)$Rejections
```

```
[1] 6
```

```
> Bayesian.SGoF(u, alpha = 0.01)$Rejections
```

```
[1] 5
```

```
> BH(u, alpha = 0.01)$Rejections
```

```
[1] 0
```

```
> BY(u, alpha = 0.01)$Rejections
```

```
[1] 0
```

Large number of tests: Hedenfalk data

As an illustrative example of a large number of dependent tests, we consider the microarray study of hereditary breast cancer of [Hedenfalk et al. \(2001\)](#). The principal aim of this study was to find genes differentially expressed between BRCA1- and BRCA2-mutation positive tumors. For that, a p value was assigned to each gene based on a suitable statistical test for the comparison. Some of them were eliminated as a result of previous analysis leaving 3170 p values. This set of p values is included in `sgof` package as `Hedenfalk`.

The first step to analyze the Hedenfalk data is to load the package and the data set. To do so, we use the next code lines:

```
> library(sgof)
> u <- Hedenfalk$x
```

Here we use the Hedenfalk data to illustrate the `BH()` and `BBSGoF()` functions which are suitable because these p values present a positive dependence ([de Uña-Álvarez, 2012](#)). We also apply to this data set the `BY()`, `SGoF()`, `Binomial.SGoF()` and `Bayesian.SGoF()` functions, and the `qvalue` procedure, to compare the results. Starting with the `BH()` function (with default argument $\alpha = 0.05$):

```
> m41 <- BH(u)
> summary(m41)
```

```
Call:
BH(u = u)
```

```
Parameters:
alpha= 0.05
```

```
$Rejections
[1] 94
```

```
$FDR
[1] 0.0356
```

```
$Adjusted.pvalues
```

```
>alpha <=alpha
 3076      94
```

The summary of the object `m41` reveals that the Benjamini and Hochberg procedure (with a FDR of 5%) is able to reject 94 null hypotheses. Next, we apply the `BY()` function with default argument $\alpha = 0.05$:

```
> m42 <- BY(u)
> summary(m42)
```

```
Call:
BY(u = u)
```

```
Parameters:
alpha= 0.05
```

```
$Rejections
[1] 0
```

```
$FDR
[1] 0
```

```
$Adjusted.pvalues
>alpha
3170
```

The output of the summary indicates that the Benjamini and Yekutieli FDR controlling procedure (with a FDR of 5%) does not declare any effect and accordingly, all the adjusted p values reported are greater than α . In fact, the smallest adjusted p value for this method is 0.0863886 ($\min(m42\$Adjusted.pvalues)$) which means that, to find at least one effect, a FDR greater than 8% should be allowed for.

In third place, we apply to the Hedenfalk data the `qvalue()` function of the **qvalue** package for comparison purposes. We obtain that the `qvalue` procedure declares 162 effects at a 5% FDR level, being clearly more powerful than the BH procedure.

```
> m43 <- qvalue(u)
> summary(m43)
```

```
Call:
qvalue(p = u)
```

```
pi0: 0.6635185
```

Cumulative number of significant calls:

	<1e-04	<0.001	<0.01	<0.025	<0.05	<0.1	<1
p-value	15	76	265	424	605	868	3170
q-value	0	0	1	73	162	319	3170

When applying the `BBSGoF()` function to the Hedenfalk data and printing the results (saved in the `m5` object), a warning alerts the user that blocks 2, 3, 4, 5, 6, 7, 8, 9, 11, 15, and 19 have been removed because they provided negative or atypical variances (see output below). We see that the `BBSGoF` procedure rejects 393 nulls. In this case, we have chosen the option `adjusted.pvalues = TRUE` in order to compute the adjusted p values with `blocks = 13` (the automatic number of blocks obtained in a preliminary application of the same function). We note that the output is not immediately obtained in this case since the computation of the adjusted p values is time-consuming. Following this, we can use again the summary method to obtain more relevant information. The summary of the `m5` object indicates that `BBSGoF`'s decision entails an estimated FDR of 12.96%. Moreover, this summary reports the automatic number of blocks, 13, corresponding to the minimum number of declared effects (searching from `kmin = 2` to `kmax = 100`), as well as the p value of the Tarone test of no correlation for this number of blocks ($5e-04$), and the parameters of the fitted beta and beta-binomial distributions.

```
> m5 <- BBSGoF(u, adjusted.pvalues = TRUE, blocks = 13)
> m5
```

```
Call:
BBSGoF(u = u, adjusted.pvalues = TRUE, blocks = 13)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
kmin= 2
kmax= 100
```

```
Warning:
Blocks 2 3 4 5 6 7 8 9 11 15 18 19 have been removed because they provided negative or
atypical variances.
```

```
Rejections:
[1] 393
> summary(m5)
```

```
...
```

```
$Rejections
[1] 393
```

```
$FDR
[1] 0.1296
```

```
$Adjusted.pvalues
>gamma <=gamma
 2777      393
```

```
$Tarone.pvalue.auto
[1] 5e-04
```

```
$beta.parameters
[1] 35.0405 148.4139
```

```
$betabinomial.parameters
[1] 0.1910 0.0054
```

```
$sd.betabinomial.parameters
[1] 0.0106 0.0038
```

```
$automatic.blocks
[1] 13
```

Figure 2 depicts the graphics obtained when using the plot method (`plot(m5)`). In the upper left plot, the p values of Tarone test are depicted. It can be seen that there are many p values falling below 0.05, thus suggesting a trend of positive correlation. In the upper right plot, the within-block correlation for each number of blocks is displayed. In the middle left plot the beta density is reported, whereas the middle right plot shows the number of effects declared for each possible number of existing blocks. The dashed line represents the number of effects declared by Conservative SGoF. Roughly, it is seen that the number of declared effects tends to increase with the number of blocks, accordingly to the weaker dependence structure. Finally, the last plot in Figure 2 represents the adjusted p values versus the original ones (for the default `adjusted.pvalues = FALSE` this last plot is not displayed).

Next, we apply the `SGoF()` function to the Hedenfalk data (with default values of `alpha` and `gamma`):

```
> m6 <- SGoF(u)
> summary(m6)
```

```
Call:
SGoF(u = u)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
```

```
$Rejections
[1] 412
```

```
$FDR
[1] 0.131
```

```
$Adjusted.pvalues
```

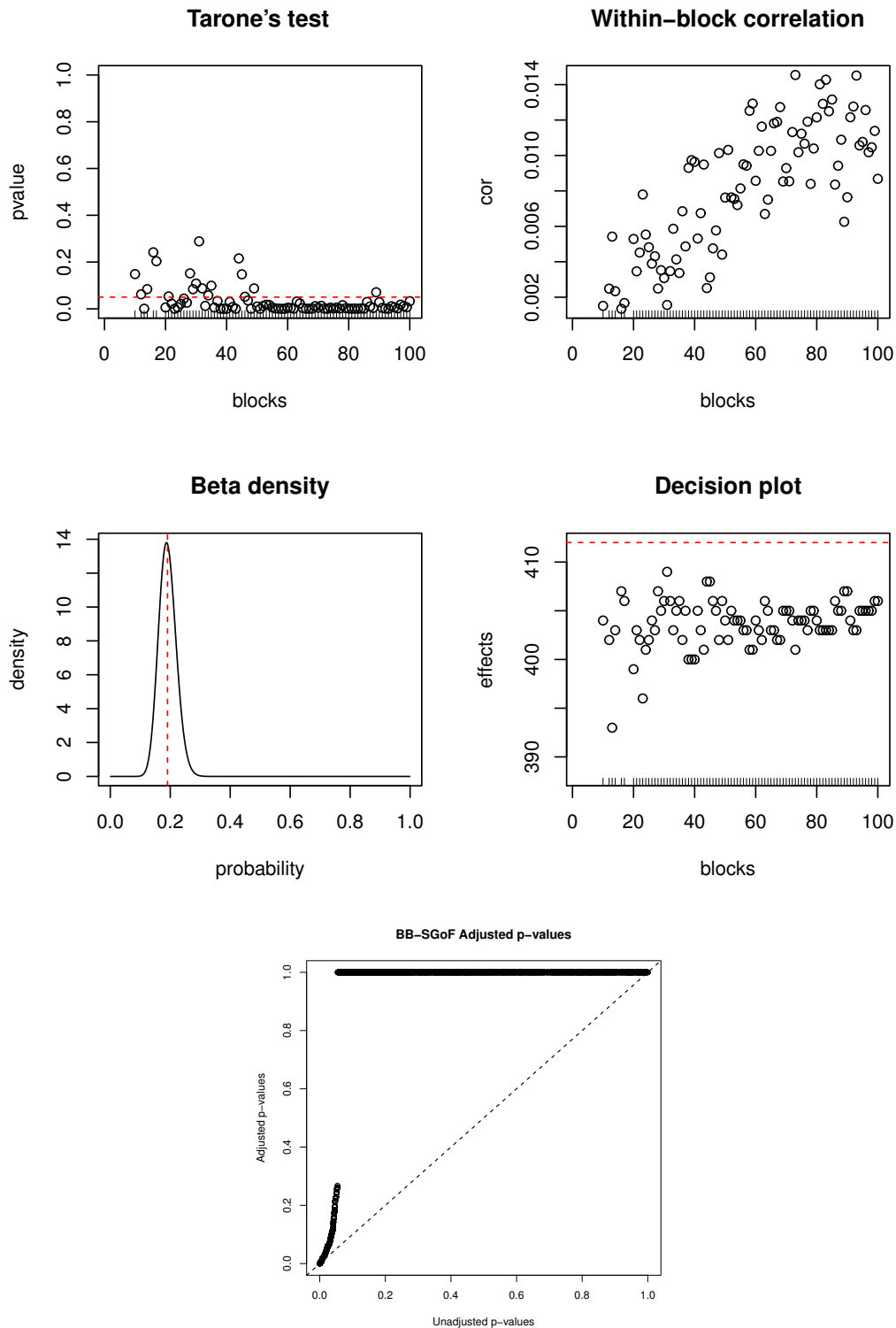


Figure 2: Hedenfalk data. Graphical results provided by the `BBSGoF()` function.

```
>gamma <=gamma
 2758   412
```

The Conservative SGoF procedure reports 412 effects with an estimated FDR of 13.1% which is a more liberal decision compared to that of BBSGoF. This is not surprising since Conservative SGoF pre-assumes independence among the tests. Figure 3 depicts the adjusted p values reported by `BH()` and `SGoF()` versus the original ones, obtained using the code lines: `plot(m41)` and `plot(m6)`.

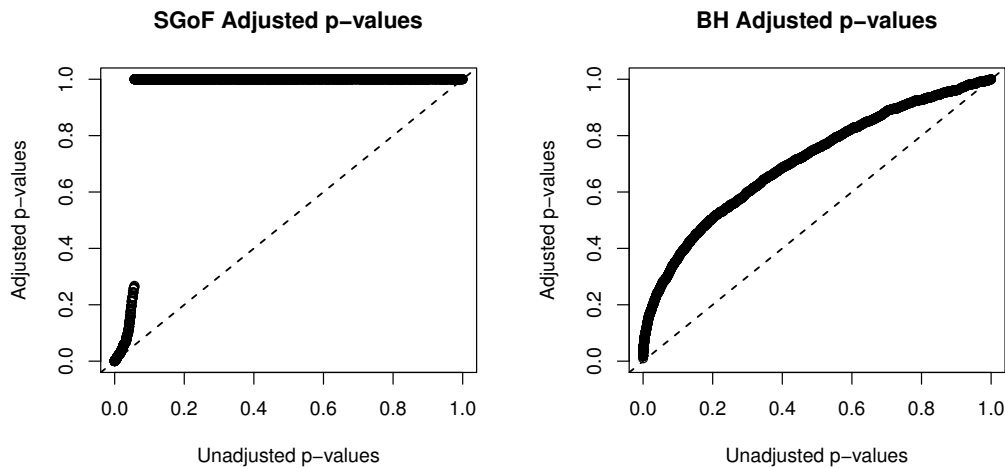


Figure 3: Hedenfalk data. Adjusted p values reported by SGoF (left) and BH (right) methods versus the original ones.

We will use the Hedenfalk data example to illustrate the role of the α and γ arguments of the SGoF-type procedures too. The `m61` object shows that Conservative SGoF with $\alpha = 0.05$ and $\gamma = 0.1$ declares 510 effects, while 520 adjusted p values are falling below γ . This illustrates how the number of rejections may change depending on the initial threshold γ . Examples below also illustrate that, when α is not equal to γ , the number of rejections may be different to the number of adjusted p values falling below γ (if $\alpha = \gamma$, the number of rejections is always a lower bound for the number of adjusted p values below γ , see [Castro-Conde and de Uña-Álvarez 2015](#); something which does not hold in general). When $\alpha = 0.1$ and $\gamma = 0.05$, SGoF() reports 420 effects, which illustrates how α has a lower impact in the Conservative SGoF procedure compared to γ . Note also that the SGoF-type methods get more liberal as the α argument increases but, when γ increases, the number of rejections may increase or decrease.

```
> m61 <- SGoF(u, gamma = 0.1)
> m61

Call:
SGoF(u = u, gamma = 0.1)

Parameters:
alpha= 0.05
gamma= 0.1

Rejections:
[1] 510

> sum(m61$Adjusted.pvalues <= m61$gamma)

[1] 520
> m62 <- SGoF(u,alpha = 0.1)
> m62

Call:
SGoF(u = u, alpha = 0.1)

Parameters:
alpha= 0.1
gamma= 0.05

Rejections:
[1] 420
> sum(m62$Adjusted.pvalues <= m62$gamma)
```

```
[1] 412
```

Finally, by applying `Binomial.SGoF()` (m7) and `Bayesian.SGoF()` (m8) functions to the Hedenfalk data one obtains 427 and 413 rejections, respectively. Binomial SGoF rejects more nulls than Conservative SGoF does (427 vs. 412), as expected, since the first method estimates the variance under the complete null of no effects. On the other hand, Bayesian SGoF reports approximately the same number of effects than Conservative SGoF, which will be generally the case with a large number of tests. Note that, as n grows, the prior information becomes less relevant and the Bayesian SGoF approaches its frequentist counterpart.

```
> m7 <- Binomial.SGoF(u)
> m7
```

```
Call:
Binomial.SGoF(u = u)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
```

```
Rejections:
[1] 427
```

```
> m8 <- Bayesian.SGoF(u)
> m8
```

```
Call:
Bayesian.SGoF(u = u)
```

```
Parameters:
alpha= 0.05
gamma= 0.05
P0= 0.5
a0= 1
b0= 1
```

```
Rejections:
[1] 413
```

Conclusions

In this paper we introduced the `sgof` package which implements in R for the first time SGoF-type multiple testing procedures; the classical FDR-controlling step-up BH and BY procedures are also included. We reviewed the definition of the several methods and discussed their relative advantages and disadvantages, and how they are implemented. Guidelines to decide which method is best suited to the data at hand have been given. Specifically, if the tests are independent, Binomial SGoF is recommended, with the possibility of using Conservative SGoF when the number of tests is moderate to large. On the other hand, BB-SGoF is suitable for serially dependent tests, while Bayesian SGoF allows for a stronger dependence structure with pairwise correlation depending on the user's a priori information. Finally, BH (independent tests or positively correlated tests) and BY (dependent tests) methods are indicated when the aim is to strongly control for the expected proportion of false discoveries. Existing improvements on BH and BY methods include the `qvalue` procedure (Storey and Tibshirani, 2003) or the empirical Bayes procedures (Pollard et al., 2005), which are implemented in other packages. `sgof` has been illustrated in practice by analyzing two real well-known data sets: Needleman data (Needleman et al., 1979) and Hedenfalk data (Hedenfalk et al., 2001). Summarizing, it has been shown that `sgof` package is very user-friendly and it is hoped that it serves the community by providing a simple and powerful tool for solving multiple testing problems.

Acknowledgements

Financial support from the Grant MTM2011-23204 (FEDER support included) of the Spanish Ministry of Science and Innovation is acknowledged.

Bibliography

- Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57(1):289–300, 1995. [p96, 97, 99]
- Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependence. *The Annals of Statistics*, 29(4):1165–1188, 2001. [p96, 99]
- J. Berger and M. Delampady. Testing precise hypotheses. *Statistical Science*, 2(3):317–352, 1987. [p98]
- J. Berger and T. Sellke. Testing a point null hypothesis: The irreconcilability of p-values and evidence. *Journal of the American Statistical Association*, 82(397):112–122, 1987. [p98]
- A. Carvajal-Rodríguez and J. de Uña-Álvarez. Assessing significance in high-throughput experiments by sequential goodness of fit and q-value estimation. *PLoS ONE*, 6(9):e24700, 2011. [p96]
- A. Carvajal-Rodríguez, J. de Uña-Álvarez, and E. Rolán-Álvarez. A new multitest correction (SGoF) that increases its statistical power when increasing the number of tests. *BMC Bioinformatics*, 10(209): 1–14, 2009. [p96, 97, 98]
- I. Castro-Conde and J. de Uña-Álvarez. SGoF multitesting method under the Bayesian paradigm. Technical Report 13/06, Statistics and OR Department, University of Vigo, 2013. URL https://webs.uvigo.es/depc05/reports/13_06.pdf. Under revision in *Statistical Applications in Genetics and Molecular Biology*. [p98]
- I. Castro-Conde and J. de Uña-Álvarez. Adjusted p-values for SGoF multiple test procedure. *Biometrical Journal*, 57(1):108–122, 2015. [p100, 110]
- I. Castro-Conde and J. de Uña-Álvarez. Power, FDR and conservativeness of BB-SGoF method. *Computational Statistics*, in press. doi: 10.1007/s00180-015-0553-2. [p96, 99]
- A. Dabney and J. D. Storey. *qvalue: Q-Value Estimation for False Discovery Rate Control*, 2014. R package version 1.40.0. [p97]
- C. Dalmaso, P. Broet, and T. Moreau. A simple procedure for estimating the false discovery rate. *Bioinformatics*, 21(5):660–668, 2005. [p100]
- J. de Uña-Álvarez. On the statistical properties of SGoF multitesting method. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–30, 2011. [p96, 98]
- J. de Uña-Álvarez. The Beta-Binomial SGoF method for multiple dependent tests. *Statistical Applications in Genetics and Molecular Biology*, 11(3), 2012. [p98, 106]
- S. Dudoit and M. J. van der Laan. *Multiple Testing Procedures with Applications to Genomics*. Springer-Verlag, 2007. ISBN: 978-0-387-49316-9. [p96, 100]
- R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5(10), 2004. [p97]
- I. Hedenfalk, D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, O. Kallioniemi, B. Wilfond, A. Borg, J. Trent, M. Raffeld, Z. Yakhini, A. Ben-Dor, E. Dougherty, J. Kononen, L. Bubendorf, W. Fehrle, S. Pittaluga, G. Gruvberger, N. Loman, O. Johannsson, H. Olsson, and G. Sauter. Gene-expression profiles in hereditary breast cancer. *New England Journal of Medicine*, 344(8):539–548, 2001. [p106, 111]
- Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4): 800–802, 1988. [p97]
- Y. Hochberg and Y. Benjamini. More powerful procedures for multiple significance testing. *Statistics in Medicine*, 9(7):811–818, 1990. [p97]
- S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2): 65–70, 1979. [p97]
- G. Hommel. A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75(2):383, 1988. [p97]

- T. Hothorn, F. Bretz, and P. Westfall. Simultaneous inference in general parametric models. *Biometrical Journal*, 50(3):346–363, 2008. [p97]
- J. T. Ladner, D. J. Barshis, and S. R. Palumbi. Protein evolution in two co-occurring types of Symbiodinium: An exploration into the genetic basis of thermal tolerance in Symbiodinium clade D. *BMC Evolutionary Biology*, 12(1):217, 2012. [p96]
- MuToss Coding Team, G. Blanchard, T. Dickhaus, N. Hack, F. Konietzschke, K. Rohmeyer, J. Rosenblatt, M. Scheer, and W. Werft. *mutoss: Unified Multiple Testing Procedures*, 2014. URL <http://CRAN.R-project.org/package=mutoss>. R package version 0.1-8. [p97]
- H. Needleman, C. Gunnoe, A. Leviton, R. Reed, H. Presie, C. Maher, and P. Barret. Deficits in psychologic and classroom performance of children with elevated dentine lead levels. *The New England Journal of Medicine*, 300(13):689–695, 1979. [p102, 111]
- T. Nichols and S. Hayasaka. Controlling the familywise error rate in functional neuroimaging: A comparative review. *Statistical Methods Medical Research*, 12:419–446, 2003. [p96]
- K. S. Pollard, S. Dudoit, and M. J. van der Laan. *Multiple Testing Procedures: R multtest Package and Applications to Genomics, in Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005. [p97, 111]
- S. Pounds and D. Fofana. *HybridMTest: Hybrid Multiple Testing*, 2011. R package version 1.10.0. [p97]
- J. Storey and R. Tibshirani. Statistical significance for genomewide studies. *Proceedings of National Academy of Sciences of the United States of America*, 100(16):9440–9445, 2003. [p100, 111]
- J. Storey, J. Taylor, and D. Siegmund. Strong control, conservative point estimation and simultaneous conservative consistency of false discovery rate: A unified approach. *Journal of the Royal Statistical Society B*, 66(1):187–205, 2004. [p97]
- J. D. Storey. The positive false discovery rate: A Bayesian interpretation and the q-value. *The Annals of Statistics*, 31(6):2013–2035, 2003. [p99]
- R. Tarone. Testing the goodness of fit of the binomial distribution. *Biometrika*, 66(3):585–590, 1979. [p99]
- G. Thompson. *SGoF Algorithm for MATLAB*. M.I.N.D. Lab, Georgia Institute of Technology and Emory University, 2010. URL http://webs.uvigo.es/acraaj/software/matlab_sgof.m. [p96]
- G. Thompson, W. Pan, M. Magnuson, D. Jaeger, and S. Keilholz. Quasi-periodic patterns (QPP): Large-scale dynamics in resting state fMRI that correlate with local infraslow electrical activity. *NeuroImage*, 84:1018–1031, 2014. [p96]
- J. Tukey. The higher criticism. *Princeton University. Course Notes, Statistics*, 411(T13), 1976. [p97]

Irene Castro-Conde
Grupo SiDOR
Facultad de Ciencias Económicas y Empresariales
Universidad de Vigo
Campus Lagoas-Marcosende
Vigo, 36310, Spain
irene.castro@uvigo.es

Jacobo de Uña-Álvarez
Departamento de Estadística e I.O.
Facultad de Ciencias Económicas y Empresariales & Centro de Investigaciones Biomédicas (CINBIO)
Universidad de Vigo
Campus Lagoas-Marcosende
Vigo, 36310, Spain
jacobo@uvigo.es

bshazard: A Flexible Tool for Nonparametric Smoothing of the Hazard Function

by Paola Rebora, Agus Salim and Marie Reilly

Abstract The hazard function is a key component in the inferential process in survival analysis and relevant for describing the pattern of failures. However, it is rarely shown in research papers due to the difficulties in nonparametric estimation. We developed the **bshazard** package to facilitate the computation of a nonparametric estimate of the hazard function, with data-driven smoothing. The method accounts for left truncation, right censoring and possible covariates. B-splines are used to estimate the shape of the hazard within the generalized linear mixed models framework. Smoothness is controlled by imposing an autoregressive structure on the baseline hazard coefficients. This perspective allows an ‘automatic’ smoothing by avoiding the need to choose the smoothing parameter, which is estimated from the data as a dispersion parameter. A simulation study demonstrates the capability of our software and an application to estimate the hazard of Non-Hodgkin’s lymphoma in Swedish population data shows its potential.

Introduction

The hazard function is the basis of the inferential process in survival analysis, and although relevant for describing the pattern of failures, is often neglected in favor of survival curves in clinical papers. The most widely applied model in survival analysis (the Cox model) allows valid comparisons in terms of hazard ratios without distributional assumptions concerning the baseline hazard function, whose nonparametric estimate is rarely shown. Thus the reference against which the relative hazard is estimated is usually ignored and a crude measure of absolute risk is sometimes provided by the Kaplan-Meier estimator that is indirectly related to the shape of the hazard function.

In the methodological literature, some methods have been developed to obtain a nonparametric hazard estimate, including kernel-based (Müller and Wang, 1994; Hess et al., 1999) and spline-based estimators (O’Sullivan, 1988; Cai et al., 2002). However, specific statistical software commands accounting for the characteristics of survival data are lacking. An exception is the R package **mu haz** (Hess and Gentleman, 2010) that estimates the hazard function from right-censored data using kernel-based methods, but this package does not accommodate left-truncated data nor does it allow for adjustment for possible covariates. Flexible parametric survival models can also be used to describe and explore the hazard function (Royston and Parmar, 2002) and in R these have been implemented in the package **flexsurv** (Jackson, 2014). In these models a transformation of the survival function is modeled as a natural cubic spline function of the logarithm of time (plus linear effects of covariates). However this approach relies on an appropriate choice of the number of knots to be used in the spline.

We have developed the **bshazard** package to obtain a nonparametric smoothed estimate of the hazard function based on B-splines and generalized linear mixed models (GLMM). This perspective enables ‘automatic’ smoothing, as the smoothing parameter can be estimated from the data as a dispersion parameter (Lee et al., 2006; Pawitan, 2001; Eilers and Marx, 1996). Our implementation accommodates the typical characteristics of survival data (left truncation, right censoring) and also possible covariates. In the following sections we briefly review the methodology and demonstrate the performance of the package in numerical examples using simulated data. We illustrate the use of the package in an application to Swedish population data, where we estimate the incidence of Non-Hodgkin’s lymphoma (NHL) in sisters of patients.

Methodological background

In this section we briefly review the methodology for smoothing the hazard rate inspired by Eilers and Marx (1996) and described in Chapter 9 of Lee et al. (2006).

Let T denote the possibly right censored failure time random variable; we are interested in the estimate of the hazard function defined as

$$h(t) = \lim_{\Delta t \rightarrow 0^+} \frac{1}{\Delta t} \cdot P(t < T \leq t + \Delta t | T \geq t)$$

so that $h(t)\Delta t$ is the probability of an event in the infinitesimal interval $(t, t + \Delta t)$, given survival to

time t .

Consider a sample of subjects that are observed from a common origin (e.g. start of exposure, time of diagnosis) to the occurrence of the event of interest. Observation can be right censored if the last follow-up time is before the occurrence of the event and/or left truncated if the observation starts after the origin (time 0). By partitioning the time axis into very small intervals, the number of events in each interval is approximately Poisson with mean $\mu(t) = h(t)P(t)$, where $P(t)$ is the total person-time in the interval (in the simplest case without censoring $P(t)$ will be the product of the number of individuals at risk at the beginning of the interval and the length of the interval). The time at risk of each subject can thus be split into n bins or time intervals (common to all subjects), similar to the life-table approach to survival analysis. For a data set with one record per individual including entry time, exit time and censoring indicator, this splitting of time can be implemented easily by the `splitLexis` function in the `Epi` package. Using (common) break points supplied by the user, the function divides each data record into disjoint follow-up intervals each with an entry time, exit time and censoring indicator, stacking these as separate ‘observations’ in the new data set. The bins should be small enough so that $h(t)$ can be considered approximately constant within each interval.

We will use t to denote the vector of the midpoints of all bins, with t_i representing the midpoint for the i^{th} bin ($i = 1, \dots, n$). The vectors $y(t)$ and $P(t)$ represent the total number of events observed and the total person-time in each interval. Using the Poisson likelihood to approximate the general likelihood for survival data (Lee et al., 2006; Lambert and Eilers, 2005; Whitehead, 1980), the hazard can be estimated by modeling the expected number of events, $\mu(t)$, in each interval as a Poisson variable by using $P(t)$ as an offset term:

$$\log[\mu(t)] = f(t) + \log[P(t)],$$

where $f(t)$ denotes the logarithm of the hazard. In this context it is straightforward to account for possible covariates in a proportional hazards scheme. After the splitting, data can be aggregated according to the bin (time) and also to the covariate values for each subject, so that the final data is organised with one record for each bin and covariate combination. Denoting by X the design matrix which contains the covariate values (fixed in time) and by β the corresponding vector of coefficients, the model becomes:

$$\log[\mu(t)] = X\beta + f(t) + \log[P(t)].$$

Note that the coefficients β do not vary with time, so that subjects with different values of the covariates X have proportional hazard rates, i.e. $\log[h(t, X = x_1)] = \log[h(t, X = x_0)] + (x_1 - x_0)\beta$.

Smoothers, such as regression splines, can be used to estimate the function $f(t) = \log[h(t)]$, and in particular B-splines provide a numerically efficient choice of basis functions (De Boor, 1978). B-splines consist of polynomial pieces of degree m , joined at a number of positions, called knots, along the time axis. The total number of knots (k) and their positions are arbitrarily chosen and are quite crucial for the final estimate, since the function can have an inflection at these locations. By using B-splines to estimate $f(t)$, the expected number of events above can be rewritten as:

$$\log[\mu(t)] = X\beta + Zv + \log[P(t)], \tag{1}$$

where Z is the matrix whose q columns are the B-splines, i.e. the values of the basis functions at the midpoints of the time bins (that will be repeated for each covariate combination) and v is a vector of length q of coefficients whose magnitude determines the amount of inflection at the knot locations. The number of basis functions $q = k + m - 1$, where k is the total number of knots, including minimum and maximum, and m is the degree of the polynomial splines. Thus the problem of estimating the hazard function reduces to the estimation of coefficients in a Generalised Linear Model framework.

B-splines are advantageous because the smoothed estimate is determined only by values at neighboring points and has no boundary effect. Nevertheless the major criticism of this approach is the arbitrary choice of the knots which determine the locations at which the smoothed function can bend and this has been subject to various investigations and discussions (Koopberg and Stone, 1992). With a small number of knots, a B-spline may not be appealing because it is not smooth at the knots (this problem may be avoided by higher-degree B-splines) and can lead to underfitting of the data. The number of knots can be increased to provide a smoother estimate, but a large number of knots may lead to serious overfitting. O’Sullivan proposed using a relatively large number of knots and preventing overfitting by a penalty term to restrict the flexibility of the fitted curve (O’Sullivan, 1986, 1988), which is analogous to likelihood-based mixed effects modeling (Eilers and Marx, 1996; Lee et al., 2006; Pawitan, 2001). In fact, the penalized least squares equation for a quadratic penalty corresponds to a mixed model log-likelihood, where the set of second-order differences of the coefficients of the B-splines (denoted Δ^2v) have autoregressive structure and are normally distributed with mean 0 and variance $\sigma_v^2 I_{q-2}$ where I_{q-2} is the identity matrix with dimension $q - 2$ and q is the number of basis functions (Lee et al., 2006).

Intuitively, since the coefficients v of the B-splines determine the change at knot locations (if the B-splines are of degree 1, they determine the change in slope) they also determine the amount of smoothing. Assuming the coefficients are normally distributed with mean 0 helps to control the amount of smoothing and has the advantage of allowing an automatic smoothing in the sense that the smoothing parameter can be estimated directly from the data as a dispersion parameter (Eilers and Marx, 1996). Thus the main criticism of the use of B-splines is overcome: the choice of knots is no longer crucial for the final estimate and in fact a large number of equally spaced knots can be chosen (more than 40 knots are rarely needed) and overfitting is prevented by the penalty (Lee et al., 2006).

More formally, for model (1) the element z_{ij} represents the value of the j^{th} basis function ($j = 1, \dots, q$) at the midpoint of the i^{th} bin ($i = 1, \dots, n$) and the $q - 2$ second-order differences of the coefficients are

$$\Delta^2 v = \begin{pmatrix} v_3 - 2v_2 + v_1 \\ v_4 - 2v_3 + v_2 \\ \dots \\ v_q - 2v_{q-1} + v_{q-2} \end{pmatrix}$$

Assuming these to be normally distributed with mean 0 and variance $\sigma_v^2 I_{q-2}$ and conditioning on these random effects, the number of observed events y is assumed to follow a Poisson distribution with overdispersion: $E(y_i|v) = \mu_i$ and $V(y_i|v) = \mu_i \phi$, where ϕ represents the dispersion parameter.

The Extended Quasi-Likelihood approximation (Lee et al., 2006) facilitates an explicit likelihood formulation (also for overdispersed Poisson data):

$$\log L(\phi, \sigma_v^2, v) = \sum \left\{ -\frac{1}{2} \log [2\pi\phi V(y(t_i))] - \frac{1}{2\phi} d[y(t_i), \mu(t_i)] \right\} - \frac{q-2}{2} \log(2\pi\sigma_v^2) - \frac{1}{2\sigma_v^2} v^T R^{-1} v,$$

where $R^{-1} = (\Delta^2(I))^T \Delta^2(I)$, A^T denotes the transpose matrix of A and $d[y(t_i), \mu(t_i)]$ is the deviance function defined by:

$$d[y(t_i), \mu(t_i)] = 2 \int_{\mu(t_i)}^{y(t_i)} \frac{y(t_i) - u}{V(u)} du.$$

This log-likelihood can be seen as a penalized likelihood where the term $v^T R^{-1} v$ is the roughness penalty; the smoothing parameter is determined by $\lambda = \frac{\phi}{\sigma_v^2}$ with a large λ implying more smoothing and a small λ denoting rough v . The extreme situation of $\lambda = 0$ corresponds to no smoothing and is analogous to consider v as a vector of fixed parameters.

The parameter ϕ , representing the dispersion parameter in the Poisson model, is usually assumed to be 1. However, if we use $\phi = 1$ when in fact $\phi > 1$ (overdispersion), we are likely to undersmooth the data, while the estimate is not influenced by underdispersion (Lee et al., 2006).

The advantage of the mixed model approach is that we have an established procedure for estimating λ , equivalent to estimating variance components in mixed models. In this setting the Iterative Weighted Least Squares (IWLS) numerical algorithm works reliably and is usually used. This algorithm uses a Taylor approximation to the extended likelihood and the application to mixed models is described in detail in Chapters 6 and 17 in Pawitan (2001).

In our application, the following iterative algorithm is used:

1. Given initial/last estimated values of λ , ϕ and $\mu(t_i)$, estimate v and β .
2. Given \hat{v} and $\hat{\beta}$, update the estimates of λ and ϕ .
3. Iterate between 1 and 2 until convergence.

We begin by implementing step 1 of the IWLS algorithm as follows. Given a fixed value of λ (starting with $\lambda = 10$ is a good starting value), we compute the working vector Y

$$Y_i = z_i^T v^0 + x_i^T \beta^0 + \log(P(t_i)) + \frac{y(t_i) - \mu(t_i)^0}{\mu(t_i)^0},$$

where z_i is the i^{th} row of Z , $y(t_i)$ and $P(t_i)$ are the number of observed events and the total person-time in the i^{th} interval and x_i denotes the i^{th} row of the matrix X_c of covariate values centered at their mean values. For the starting values $\mu(t_i)^0$ we take the average over all time intervals of the raw hazard, computed as the number of events divided by the person-time at risk in the interval, $\frac{y(t_i)}{P(t_i)}$. As for the coefficients, we take $v^0 = \log[\mu(t)^0]$ and $\beta^0 = 0$. Defining W as the variance of the working vector (Y) with elements $w_i = \mu(t_i)^0$, the updating formula for the random effects is the solution to

$$\left(Z^T W Z + \lambda R^{-1} \right) v = Z^T W (Y - \log(P(t)) - X_c \beta)$$

where $R^{-1} = (\Delta^2(I))^T \Delta^2(I)$ and β is the solution to

$$\left(X_c^T W X_c \right) \beta = X_c^T W (Y - \log(P(t)) - Zv).$$

Note that if λ is set to 0, v is estimated as a vector of fixed parameters as mentioned above.

At this point we can introduce a quantity to describe the complexity of the smoothing, that is the effective number of parameters (or degrees of freedom) associated with v , denoted by df and given by:

$$df = \text{trace} \left\{ \left(Z^T W Z + \lambda R^{-1} \right)^{-1} Z^T W Z \right\}.$$

When λ is set to 0, df is equal to the number of basis functions q , while it decreases with increasing penalisation.

For step 2, given \hat{v} and $\hat{\beta}$, the dispersion parameter is updated by the method of moments (Wedderburn, 1974) as follows:

$$\hat{\phi} = \text{var} \left[\frac{y(t_i) - \widehat{\mu}(t_i)}{\sqrt{\widehat{\mu}(t_i)}} \right].$$

An estimated variance greater than 1 would suggest overdispersion in the data. When we believe overdispersion is not present (ϕ close to 1), we suggest fixing ϕ at 1, especially when adjusting for covariates where there is a greater risk of overfitting.

The quantity σ_v^2 can be updated (Lee et al., 2006) by:

$$\hat{\sigma}_v^2 = \frac{\hat{v}^T R^{-1} \hat{v}}{df - 2}.$$

Once convergence is reached, a point-wise confidence band for the smooth estimate is computed for the linear predictor $\log[\hat{\mu}(t)]$ with variance matrix (assuming the fixed parameters are known at the estimated values) $H = Z(Z^T W Z + \lambda R^{-1})^{-1} Z^T$. This is then transformed to the hazard scale by:

$$\frac{\exp \left\{ \log[\hat{\mu}(t)] \pm z_{\alpha/2} \sqrt{H} \right\}}{P(t)},$$

where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ percentile of a standard normal distribution.

Getting started

This section provides explanations of the input data and output data of the package **bshazard**, which estimates the hazard function nonparametrically. In order to use package **bshazard**, the following R packages need to be installed: **survival**, **Epi** and **splines** (Therneau, 2014; Carstensen et al., 2011). After installation, the main function can be called by:

```
library(bshazard)
bshazard(formula, data, nbin, nk, degree, l0, lambda, phi, alpha, err, verbose)
```

The only mandatory argument is `formula`, which should contain a survival object (interpreted by the **survival** package) with the time of event (possibly in a counting process format), the censoring indicator and (optionally) the covariates. For example, the function

```
output <- bshazard(formula = Surv(time_of_entry, time_of_exit,
  censoring_indicator ~ covariates))
```

will produce a smoothed estimate of the hazard accounting for covariates with the following default settings:

- the time axis split at each distinct observed time in the data (censoring or event);
- B-splines with 31 knots;
- degree of B-splines 1;
- smoothing parameter estimated from the data (with starting value 10);
- overdispersion parameter estimated from the data;
- 95% confidence intervals.

By providing various arguments in the function call, the user can override the default settings for the data format (e.g. specify a data frame), the number of bins (`nbin`), the number of knots (`nk`), degree of splines (`degree`), smoothing and overdispersion parameters (`lambda` and `phi`), confidence level (`alpha`) and convergence criterion (`err`). Detailed explanations are provided in the function's help file.

The output of the `bshazard` function includes a data frame with the original data split in time intervals (`raw.data`), vectors containing the estimated hazard and confidence limits and the parameter estimates (coefficients of the covariates, $\hat{\phi}$, $\hat{\sigma}_\sigma^2$, df).

The package also includes the following functions:

- `summary(output)` prints the values of the estimated hazard, with confidence limits, for each observed time and the parameter estimates;
- `plot(output)` and `lines(output)` plot the hazard, with confidence limits.

Numerical examples

In order to test the flexibility of the proposed algorithm we simulated data from a non monotone function that could represent, for example, the seasonality of flu incidence:

$$h(t) = b \cdot \left[h_1^{p_1} p_1 t^{p_1-1} e^{-(h_1 \cdot t)^{p_1}} \right] + (1 - b) \cdot \left[h_2^{p_2} p_2 t^{p_2-1} e^{-(h_2 \cdot t)^{p_2}} \right], \quad (2)$$

where $h(t)$ is the hazard function, b is a Bernoulli random variable with parameter 0.6, $h_1 = 1.2$, $p_1 = 2$, $h_2 = 0.3$, $p_2 = 5$. The choice of the parameter values was inspired by [Cai et al. \(2002\)](#). We considered samples of 500 subjects and for each subject we also simulated a censoring time as $U(0, 5)$. Under this model we simulated 500 random samples and, for each sample, we estimated the hazard function by:

```
fit <- bshazard(Surv(exit_time, cens) ~ 1, data = dati, nbin = 100)
```

The choice to pre-bin the data in 100 time intervals was for simplicity of comparison of different estimates of hazard from different simulations at the same time point. The hazard function estimate did not change when using different numbers of bins or different numbers of knots (data not shown). The results of this simulation are summarised in [Figure 1](#). The mean estimate of the hazard function is very close to the true hazard. For comparison, we also estimated the hazard using `muhaz(exit_time, cens, max.time = 3, bw.method = "g", n.est.grid = 100)` and plotted its mean estimate in [Figure 1](#), where it can be seen to be very close to the true hazard. Under the same distribution we also simulated a left truncation time l as $U(-1, 1)$, with $l < 0$ considered as 0 (late entry for half of the subjects). In this simulation, only subjects with event/censoring times greater than the left truncation time were valid for analysis. This setting provided results very similar to the previous setting (data not shown).

In a second simulation, we included a covariate X generated as a standard normal random variable that influenced the hazard rate according to the model:

$$h(t) = 0.5t \cdot \exp(X).$$

Under this model we again simulated 500 random samples and, for each sample, estimated the hazard function by:

```
fit <- bshazard(Surv(entry_time, exit_time, cens) ~ x, data = dati, nbin = 30)
```

Results are shown in [Figure 2](#). The mean estimate of the hazard function is again very close to the true hazard. Note that when adjusting for continuous covariates, the hazard is estimated under the assumption of a linear effect of the covariates centered at their mean values. In this case, a comparison with the `muhaz` function was not possible given that it does not accommodate covariates or late entry.

Results from non-Hodgkin's lymphoma data

In this section we use the `bshazard` package to estimate the hazard of cancer diagnosis using data from Swedish population registers. For all individuals born in Sweden since 1932, the Multi-Generation Register maintained by Statistics Sweden ([Statistics Sweden, 2009](#)) identifies the biological parents, thus enabling the construction of kinships within families. Using the individually unique national registration number, individuals in this register can be linked to the National Cancer Register, which records all malignancies diagnosed since 1958. In our application we use data from [Lee et al. \(2013\)](#) who analyzed 3,015 sisters of 1,902 non-Hodgkin's lymphoma (NHL) female patients and 15,697 sisters of 3,836 matched controls who were cancer free at the time of diagnosis of the corresponding case and who matched the case with respect to age, year of birth, sex and county of residence.

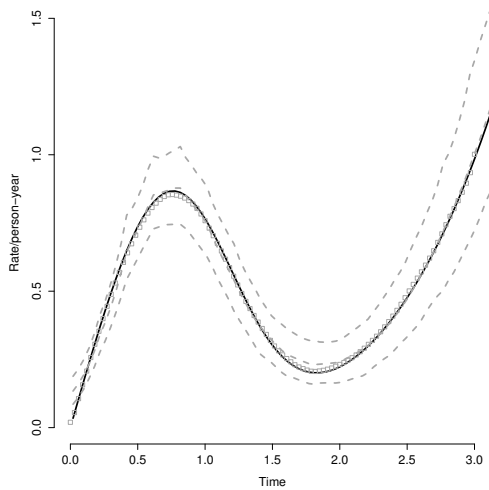


Figure 1: Numerical example; estimated and true hazard function ($h(t) = b[1.2^{22}te^{-(1.2t)^2}] + (1 - b)[0.3^5 5t^4 e^{-(0.3t)^5}]$) with right censoring for $n = 500$. The solid black line is the true hazard function, the dashed gray lines are the mean estimate from bshazard and the empirical pointwise 95% confidence interval. Squares represent the mean estimate from the muhaz function.

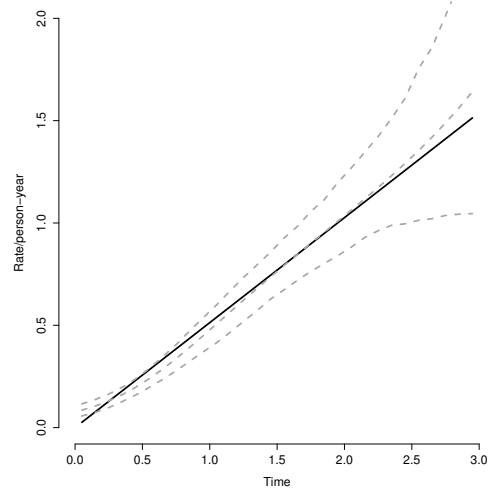


Figure 2: Numerical example; estimated and true hazard function ($h(t) = 0.5t \exp(X)$) with left truncation and right censoring for $n = 500$. The solid black line represents the true hazard function, and the dashed gray lines are the mean estimate and the empirical pointwise 95% confidence interval.

Sisters are at risk from birth to the age at NHL diagnosis or censoring due to death, emigration or end of study (2007). Individuals born before the start of the cancer register (1958) were considered at risk from their age at the start of the cancer register, resulting in delayed entry. In the study period, 32 NHL diagnoses were observed in the exposed group (sisters of subjects with a diagnosis of NHL) and 39 in the unexposed group (sisters of cancer-free subjects).

In order to illustrate the automatic smoothing, we first concentrated on sisters of cancer-free subjects and computed the smoothed hazard using the usual B-splines (i.e. setting the smoothing parameter $\lambda = 0$):

```
fit_notexp_l0 <- bshazard(Surv(entry_age, exit_age, cens) ~ 1, lambda = 0,
  data = sis[nexpo])
```

where [nexpo] selects only the sisters of control subjects.

The resulting hazard function, plotted in Figure 3, has several bumps, so we proceeded to estimate the smoothing parameter from the data, again using the same number of knots (31 as default):

```
fit_notexp <- bshazard(Surv(entry_age, exit_age, cens) ~ 1, data = sis[nexpo])
```

The resulting estimate of λ was 11,311.84 with 2.40 degrees of freedom and $\hat{\phi} = 0.82$ and the dotted line in Figure 3 presents the corresponding hazard estimate. The estimate of the hazard function was unchanged when using different numbers of knots or by setting the overdispersion parameter to 1 (in fact no overdispersion was found $\hat{\phi} = 0.82$).

Lee et al. found that sisters of female NHL patients have hazard ratio of NHL of 4.36 (95% confidence interval [2.74; 6.94]) compared to sisters of controls (Lee et al., 2013). For comparison with their results we estimated the risk of NHL adjusting for ‘exposure’ (i.e. being a sister of a case rather than a control):

```
fit_adj_exp <- bshazard(Surv(entry_age, exit_age, cens) ~ case, data = sis)
```

where the variable case indicates whether the subject is a sister of a case. We obtained a very similar hazard ratio, $\exp(\hat{\beta}) = 4.35$, as expected. Note that the code provides the hazard for a subject with covariate value equal to the mean of all subjects:

$$\hat{h}(t; \bar{x}) = \frac{\hat{\mu}(t, \bar{x})}{P(t)} = \frac{\exp[\bar{x}\hat{\beta} + z\hat{v} + \log(P(t))]}{P(t)}$$

Since this estimate is not meaningful for categorical variables, we obtained separate hazard

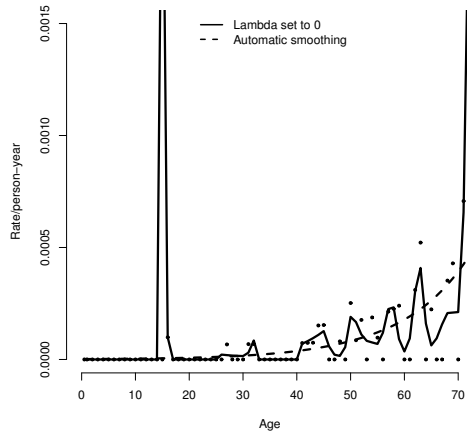


Figure 3: Hazard estimate of NHL in sisters of controls with smoothing parameter set to 0 (continuous line) and smoothing estimated from the data (dashed line). Dots represent the raw hazard.

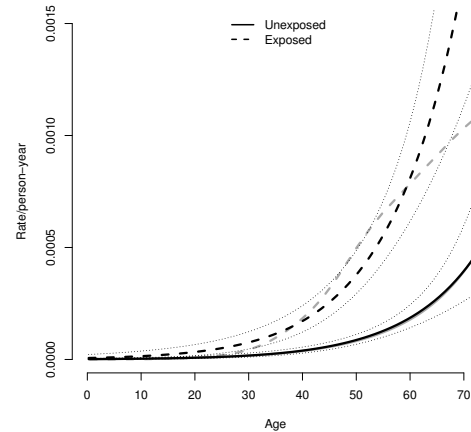


Figure 4: Smoothed hazard of NHL in unexposed (continuous line) and exposed (dashed line) sisters obtained from the model with exposure as a covariate. Dotted lines represent confidence intervals. For reference, the stratified estimates obtained from separate models for exposed and unexposed sisters are presented in grey.

estimates for unexposed and exposed subjects from:

$$\hat{h}(t; x) = \frac{\exp [x\hat{\beta} + z\hat{v} + \log(P(t))]}{P(t)} = \hat{h}(t; \bar{x}) \cdot \exp [(x - \bar{x})\hat{\beta}].$$

The function `plot` in the package `bshazard` calls an object of class ‘bahazard’ and allows one to easily plot separate curves for each set of covariate values. The estimates plotted in Figure 4 were obtained using `plot(fit_adj_exp, overall = FALSE)` and assume proportionality of the hazard in exposed and unexposed sisters. As a reference, we also performed stratified analyses obtaining estimates separately for exposed and unexposed sisters and these are plotted in grey in Figure 4. As expected, the hazard estimates are similar to the separate estimates, but are constrained by the assumption of proportionality. This is especially evident in the exposed group: the hazard increase seems to level off after age 55, but this is not detected by the hazard estimate obtained under the joint adjusted model.

The adjustment to the hazard estimate is particularly advantageous for continuous variables. The proposed method allows inclusion of more than one covariate, so in the NHL application the hazard of exposed and unexposed subjects could be further adjusted for calendar time by:

```
fit_adj_exp_caly <- bshazard(Surv(entry_age, exit_age, cens) ~ case + yob, data = sis)
```

This yielded hazard estimates that were essentially unchanged and are not reported here.

Discussion

We have implemented a software package in R to obtain a nonparametric smoothed estimate of the hazard function based on B-splines from the perspective of generalized linear mixed models. The Poisson model leads to an elegant form for the log of the hazard (Lambert and Eilers, 2005). We adopted the discrete approach to survival analysis allowing staggered entry, intermittent risk periods and large data sets to be handled with ease. The code is easy to use and accommodates the typical characteristics of survival data (left truncation, right censoring) by using the `survival` package. It also accounts for possible covariates, but the user should be aware that covariates are assumed to have a constant effect on the hazard (proportional hazards). It is of note that the model is also valid for estimating the rate function over time where an individual can have repeated events (Cook and Lawless, 2002). For such applications, the code can be used without change and data should be included in a counting process format. The package `bshazard` is available from <http://CRAN.R-project.org/package=bshazard>.

The main advantage of our function is that the user can obtain the estimated hazard by a simple line of code and that the extent of smoothing is data-driven (i.e. the user does not need to specify any smoothing parameter). The number of degrees of freedom gives an intuitive interpretation of the amount of smoothing and is also useful for making comparisons between different smoothers.

To prepare the data for analysis the package uses the `splitLexis` function (**Epi** package). The choice of time intervals does not affect the smoothed estimate as long as the bins are small enough for the assumption of constant hazard within the bin to be reasonably satisfied. For large numbers of observations the splitting can be time consuming, especially when accounting for many covariates. Nevertheless, in our relatively large data-set of NHL sisters, the most complex hazard estimate, adjusting for two covariates (`fit_adj_exp_caly`) was obtained in less than one minute. Interval censored data are not included in the code at this time, but the package can still be used if the censored intervals are relatively short. In this situation we could choose the bins in such a way that every censored interval is completely included in one bin, avoiding the problem of the specification of the exact event time, but some assumptions on person-time at risk will be needed. With small data sets and in the presence of covariates, estimation of both smoothing and overdispersion parameters can cause some convergence problem; in this case if there is not strong evidence of overdispersion we suggest fixing ϕ at 1.

The possibility to estimate the hazard function with a simple command provides a useful tool for a deeper understanding of the process being studied, both in terms of the magnitude of the risk and the shape of the curve (Rebora et al., 2008). For example, in a previous paper, we found that sisters of subjects diagnosed with NHL have a hazard ratio of 4.36 (95% confidence interval [2.74; 6.94]) for NHL compared to sisters of controls (Lee et al., 2013), but did not show at which age the risk was higher. Reanalyzing the data using **bshazard** revealed how the magnitude of the risk varied with age. This important information is often neglected in epidemiological studies, in large part due to the lack of simple and accessible software tools. An important area of application is in the presence of time-dependent variables, when an absolute measure of risk cannot be obtained by the Kaplan-Meier estimator. For example, in a comparison between the effect on disease-free survival of chemotherapy and transplantation, which occur at different time points, the Kaplan-Meier method will tend to overestimate the survival of the transplanted group, since these patients have to survive until transplant (immortal time bias). In such situations, a hazard estimate is particularly useful for presenting the instantaneous risk of an event over time given that it conditions on the subjects at risk at each time.

In summary, the **bshazard** package can enhance the analysis of survival data in a wide range of applications. The advantage of automatic smoothing and the close relationship with the `survfit` function make the package very simple to use.

Acknowledgments

The authors would like to thank Yudi Pawitan who provided the initial code for smoothing.

Bibliography

- T. Cai, R. J. Hyndman, and M. P. Wand. Mixed model-based hazard estimation. *Journal of Computational and Graphical Statistics*, 11(4):784–798, 2002. [p114, 118]
- B. Carstensen, M. Plummer, E. Laara, and M. H. e. al. *Epi: A Package for Statistical Analysis in Epidemiology*, 2011. URL <http://CRAN.R-project.org/package=Epi>. R package version 1.1.20. [p117]
- R. J. Cook and J. F. Lawless. Analysis of repeated events. *Statistical Methods in Medical Research*, 11(2): 141–166, 2002. [p120]
- C. De Boor. *A Practical Guide to Splines*. New York: Springer Verlag, 1978. [p115]
- P. H. C. Eilers and B. D. Marx. Flexible Smoothing with B-splines and Penalties. *Statistical Science*, 11 (2):89–102, 1996. [p114, 115, 116]
- K. Hess and R. Gentleman. *mu haz: Hazard Function Estimation in Survival Analysis*, 2010. URL <http://CRAN.R-project.org/package=mu haz>. R package version 1.2.5. [p114]
- K. R. Hess, D. M. Serachitopol, and B. W. Brown. Hazard function estimators: A simulation study. *Statistics in Medicine*, 18(22):3075–3088, 1999. [p114]

- C. Jackson. *flexsurv: Flexible Parametric Survival and Multi-State Models*, 2014. URL <http://CRAN.R-project.org/package=flexsurv>. R package version 0.5. [p114]
- C. Kooperberg and C. J. Stone. Log-spline density estimation for censored data. *Journal of Computational and Graphical Statistics*, 1(4):301–328, 1992. [p115]
- P. Lambert and P. H. C. Eilers. Bayesian proportional hazards model with time-varying regression coefficients: A penalized poisson regression approach. *Statistics in Medicine*, 24(24):3977–3989, 2005. [p115, 120]
- M. Lee, P. Rebora, M. G. Valsecchi, K. Czene, and M. Reilly. A unified model for estimating and testing familial aggregation. *Statistics in Medicine*, 32(30):5353–5365, 2013. [p118, 119, 121]
- Y. Lee, J. A. Nelder, and Y. Pawitan. *Generalized Linear Models with Random Effects: Unified Analysis via H-Likelihood*, volume 106. Chapman & Hall/CRC, 2006. [p114, 115, 116, 117]
- H.-G. Müller and J.-L. Wang. Hazard rate estimation under random censoring with varying kernels and bandwidths. *Biometrics*, 50(1):61–76, 1994. [p114]
- F. O’Sullivan. A statistical perspective on ill-posed inverse problems. *Statistical Science*, 1(4):502–518, 1986. [p115]
- F. O’Sullivan. Fast computation of fully automated log-density and log-hazard estimators. *SIAM Journal on Scientific and Statistical Computing*, 9(2):363–379, 1988. [p114, 115]
- Y. Pawitan. *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford University Press, 2001. [p114, 115, 116]
- P. Rebora, K. Czene, and M. Reilly. Timing of familial breast cancer in sisters. *Journal of the National Cancer Institute*, 100(10):721–727, 2008. [p121]
- P. Royston and M. K. B. Parmar. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine*, 21(15):2175–2197, 2002. [p114]
- Statistics Sweden. *The Multi-Generation Register 2008. A Description of Contents and Quality*, 2009. [p118]
- T. Therneau. *survival: A Package for Survival Analysis in S*, 2014. URL <http://CRAN.R-project.org/package=survival>. R package version 2.37-7. [p117]
- R. W. M. Wedderburn. Quasi-likelihood functions, generalized linear models and the Gauss-Newton method. *Biometrika*, 61(3):439–447, 1974. [p117]
- J. Whitehead. Fitting Cox’s regression model to survival data using GLIM. *Applied Statistics*, 29(3): 268–275, 1980. [p115]

Paola Rebora
Department of Health Sciences
University of Milano-Bicocca
Via Cadore 48 20900 Monza, Italy
paola.rebora@unimib.it

Agus Salim
Department of Mathematics & Statistics
La Trobe University
Bundoora, Australia
a.salim@latrobe.edu.au

Marie Reilly
Department of Medical Epidemiology and Biostatistics
Karolinska Institutet
BOX 281, 171 77 Stockholm, Sweden
marie.reilly@ki.se

SMR: An R Package for Computing the Externally Studentized Normal Midrange Distribution

by Ben Dêivide Oliveira Batista and Daniel Furtado Ferreira

Abstract The main purpose of this paper is to present the main algorithms underlining the construction and implementation of the **SMR** package, whose aim is to compute studentized normal midrange distribution. Details on the externally studentized normal midrange and standardized normal midrange distributions are also given. The package follows the same structure as the probability functions implemented in R. That is: the probability density function (dSMR), the cumulative distribution function (pSMR), the quantile function (qSMR) and the random number generating function (rSMR). Pseudocode and illustrative examples of how to use the package are presented.

Introduction

The **SMR** package was created to provide an infrastructure for the studentized midrange distribution. This is a new distribution that was inspired by the externally studentized range distribution, which has been largely applied in multiple comparison procedures to identify the best treatment level and has been extensively studied theoretically. Several algorithms to compute the probability density, cumulative distribution, and quantile functions were published by [Lund and Lund \(1983\)](#) and [Copenhaver and Holland \(1988\)](#). Recently, [Batista and Ferreira \(2014\)](#) developed the theory of the externally studentized normal midrange distribution, which is new in the scientific literature to the best knowledge of the authors. The cumulative distribution, the probability density, and the quantile functions were obtained by them analytically.

Computations of the required multidimensional integrations should be done numerically. Therefore, [Batista and Ferreira \(2014\)](#) applied Gaussian quadrature for this task. In particular, they chose the Gauss-Legendre quadrature for solving numerical integrations, because it obtains more accurate results when compared with other Gaussian quadrature methods. The quantile function of the externally studentized normal midrange was computed by the Newton-Raphson method. Based on these numerical methods, the **SMR** package was built and released.

The package name was chosen to identify the Studentized MidRange distribution. The package follows the same structure as the probability functions implemented in R. The following functions were implemented in the package: the probability density function (dSMR), the cumulative distribution function (pSMR), the quantile function (qSMR), and the random number generating function (rSMR).

Therefore, the main purpose of this paper is to present the main algorithms underlining the construction and implementation of the **SMR** package, showing pseudocode of its functions and providing the fundamental ideas for the appropriate use of the package through illustrative examples.

First, details on the externally studentized normal midrange and standardized normal midrange distributions are given. Second, the algorithms for the construction of the package and their respective pseudocodes are presented. Third, details of the **SMR** package functions are showed. Finally, illustrative examples of the package are presented.

The externally studentized normal midrange distribution

Let $X_{1,n}, X_{2,n}, \dots, X_{n,n}$ be the order statistics of a random sample X_1, X_2, \dots, X_n of size n from a population with cumulative distribution function (c.d.f.) $F(x)$ and probability density function (p.d.f.) $f(x)$ arranged in order of magnitude of values.

The midrange R for a sample X_1, X_2, \dots, X_n is defined as

$$R = (X_{1,n} + X_{n,n})/2,$$

see, e.g., [Rider \(1957\)](#).

The p.d.f. and c.d.f. of R are

$$f_R(r) = \int_{-\infty}^r 2n(n-1)f(y)f(2r-y)[F(2r-y) - F(y)]^{n-2}dy \quad (1)$$

and

$$F_R(r) = n \int_{-\infty}^r f(y)[F(2r - y) - F(y)]^{n-1} dy, \tag{2}$$

see, e.g., [David and Nagaraja \(2003\)](#).

The studentized midrange Q for a sample X_1, X_2, \dots, X_n is defined as

$$Q = \frac{R/\sigma}{S/\sigma} = \frac{R}{S} = \frac{W}{X},$$

see, e.g., [Batista \(2012\)](#). When S and R are independent, Q corresponds to the externally studentized midrange, otherwise Q is the internally studentized midrange. The former occurs for example, when R is computed from one normal random sample of size n , and S is obtained from another random sample of size m . It also occurs when R/S is computed from the treatment means of an experimental factor with n levels and $S = \sqrt{(QME)}$, where QME is the associated experimental mean square error with $m - 1$ degrees of freedom ([Searle, 1987](#)). When considering the externally studentized midrange, it is relevant to compute Q as the ratio of $W = R/\sigma$ the standardized midrange distribution and $X = S/\sigma$.

Considering the particular case of the normal distribution with mean $\mu = 0$, without loss of generality, and variance σ^2 the p.d.f. of $X = S/\sigma$ is given by ([Newman, 1939](#))

$$f(x; \nu) = \frac{\nu^{v/2}}{\Gamma(\nu/2)2^{\nu/2-1}} x^{\nu-1} e^{-\nu x^2/2}, x \geq 0. \tag{3}$$

An important distribution for this study, well documented in [Batista \(2012\)](#), [Gumbel \(1958\)](#) and [Pillai \(1950\)](#), is the standardized normal midrange distribution, defined by $W = R/\sigma$. The p.d.f. of W is

$$f_W(w) = \int_{-\infty}^w 2n(n - 1)\phi(y)\phi(2w - y)[\Phi(2w - y) - \Phi(y)]^{n-2} dy. \tag{4}$$

and the c.d.f. is

$$F_W(w) = \int_{-\infty}^w n\phi(y)[\Phi(2w - y) - \Phi(y)]^{n-1} dy, \tag{5}$$

both results found in [David and Nagaraja \(2003\)](#), where $\Phi(\cdot)$ and $\phi(\cdot)$ are the c.d.f. and p.d.f. of the standard normal distribution $N(0, 1)$, respectively.

Therefore, according to [Batista and Ferreira \(2014\)](#), the p.d.f. and c.d.f. of Q , in the particular case of the normal distribution with mean $\mu = 0$ and variance σ^2 , are respectively

$$f(q; n, \nu) = \int_0^\infty \int_{-\infty}^{xq} 2n(n - 1)x\phi(y)\phi(2xq - y)[\Phi(2xq - y) - \Phi(y)]^{n-2} f(x; \nu) dy dx, \tag{6}$$

and

$$F(q; n, \nu) = \int_0^\infty \int_{-\infty}^{xq} n\phi(y)[\Phi(2xq - y) - \Phi(y)]^{n-1} f(x; \nu) dy dx, \tag{7}$$

where ν is the number of degrees of freedom.

The p.d.f (4) and c.d.f. (5) are very important to the externally studentized normal midrange algorithms implementation in the **SMR** package.

Algorithms used in the SMR package

The functions implemented in the **SMR** package are dependent on specific functions of R which are: `pnorm`, to obtain the cumulative distribution function of the standard normal, $\Phi(x)$; `dnorm`, to obtain the standard normal probability density function, $\phi(x)$; `lgamma`, to obtain the logarithm of the gamma function. To compute the nodes and weights of the Gauss-Legendre quadrature, an R function based on the method presented by [Hildebrand \(1974\)](#) was implemented. In the following subsections the algorithms and pseudocodes used in the construction of each routine of **SMR** package are presented.

The Gauss-Legendre quadrature and the Newton-Raphson method

The basic idea of Gauss-Legendre quadrature of a function $f(x)$, according to [Davis and Rabinowitz \(1984\)](#), is to write it as follows:

$$\int_{-1}^1 f(x)dx = \int_{-1}^1 w(x)g(x)dx \approx \sum_{i=1}^m w_i g(x_i), \tag{8}$$

where $f(x) \equiv w(x)g(x)$ and $w(x)$ is the weight function in the Gaussian quadrature, w_i and x_i are the nodes and weights, respectively, in an m -point Gaussian quadrature rule, for $i = 1, 2, \dots, m$. The weight function is $w(x) = 1$ in the Gauss-Legendre quadrature, thus $f(x) = g(x)$.

Let the symmetric tridiagonal matrix \mathcal{J} , with elements $\mathcal{J}_{i,i} = a_{i-1}$, $i = 1, \dots, m$ and $\mathcal{J}_{i-1,i} = \mathcal{J}_{i,i-1} = \sqrt{J_{i,i-1}J_{i-1,i}} = \sqrt{b_{i-1}}$, $i = 2, \dots, m$, be the Jacobi matrix, given by

$$\mathcal{J} = \begin{pmatrix} a_0 & \sqrt{b_1} & 0 & \dots & \dots & \dots \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & 0 & \dots & \dots \\ 0 & \sqrt{b_2} & a_2 & \sqrt{b_3} & 0 & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & 0 & \sqrt{b_{m-2}} & a_{m-2} & \sqrt{b_{m-1}} \\ \dots & \dots & \dots & 0 & \sqrt{b_{m-1}} & a_{m-1} \end{pmatrix},$$

where $a_{i-1} = 0$, for $i = 1, \dots, m$ and $\sqrt{b_{i-1}} = (i-1)/\sqrt{4(i-1)^2-1}$, for $i = 2, \dots, m$, considering the Gauss-Legendre quadrature.

For computing the quadrature weights and nodes, the eigenvalues and eigenvectors of \mathcal{J} were obtained [Hildebrand \(1974\)](#). The nodes for the Gaussian quadrature were the eigenvalues of this tridiagonal matrix \mathcal{J} and the weights can be computed from the corresponding normalized eigenvectors $\phi^{(i)}$ associated to the eigenvalue x_i . The weight w_i can be computed from the first element $\phi_1^{(i)}$ of the i th corresponding eigenvector by

$$w_i = \mu_0 \left(\phi_1^{(i)} \right)^2,$$

where $\mu_0 = 2$ for the Gauss-Legendre quadrature ([Gil et al., 2007](#)).

The set $\{x_i, w_i\}$ is determined such that expression (8) yields an exact result for polynomials of degree $2m - 1$ or less ([Chihara, 1978](#)). For non-polynomial functions the Gauss-Legendre quadrature error is defined by

$$\varepsilon_m \approx \left| \sum_{i=1}^q w_i g(x_i) - \sum_{i=1}^m w_i g(x_i) \right|, \quad q > m. \tag{9}$$

However, the externally studentized normal midrange distribution depends on integrals over infinite intervals. The integral over an infinite range should be changed into an integral over $[-1,1]$ by using the following transformations ([Davis and Rabinowitz, 1984](#))

$$\int_a^\infty f(y)dy = \int_{-1}^1 f\left(a + \frac{1+t}{1-t}\right) \frac{2}{(1-t)^2} dt, \tag{10}$$

$$\int_{-\infty}^b f(y)dy = \int_{-1}^1 f\left(b + \frac{1+t}{t-1}\right) \frac{2}{(t-1)^2} dt, \tag{11}$$

$$\int_a^b f(y)dy = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{(b-a)}{2}t + \frac{(b+a)}{2}\right) dt. \tag{12}$$

Therefore, the integrals were computed by applying the Gauss-Legendre quadrature rule on these

transformed variables by

$$\int_a^\infty f(y)dy \approx \sum_{i=1}^m w_i \frac{2}{(1-t_i)^2} f\left(a + \frac{1+t_i}{1-t_i}\right), \tag{13}$$

$$\int_{-\infty}^b f(y)dy \approx \sum_{i=1}^m w_i \frac{2}{(t_i-1)^2} f\left(b + \frac{1+t_i}{t_i-1}\right), \tag{14}$$

$$\int_a^b f(y)dy \approx \frac{b-a}{2} \sum_{i=1}^m w_i f\left(\frac{b-a}{2}t_i + \frac{a+b}{2}\right). \tag{15}$$

For more details, see [Olver et al. \(2010\)](#).

The Newton-Raphson approximation aims to find the roots of a real function, that is,

$$y : f(y) = 0.$$

Thus, iteratively, the Newton-Raphson method follows

$$y_{k+1} = y_k - \frac{f(y_k)}{f'(y_k)}, \quad k = 0, 1, 2, \dots$$

where $f'(y)$ is the first derivative of the function $f(y)$, see, e.g., [Miller \(2014\)](#).

The process starts with an initial arbitrary value y_0 , which should be reasonably close to the true root, then the method will usually converge in a few iterations, considering $f'(y_k) \neq 0$.

In this study the main objective was to find the quantile q , given that the cumulative probability is p , $0 < p < 1$. Hence, considering the cumulative distribution function, $F(q)$, and the probability density, $f(q)$, functions of the random variable Q , the desired solution is

$$q : F(q) - p = 0.$$

The solution is obtained by

$$q_{k+1} = q_k - \frac{F(q_k) - p}{f(q_k)}, \quad k = 0, 1, 2, \dots \tag{16}$$

that should be computed sequentially until a certain convergence criterion is reached.

Standardized normal midrange probability density function

For the standardized normal midrange probability density function (4), the integration interval $(-\infty; q]$ was divided into two subintervals and w was replaced in the algorithm to q , as done before to the cumulative distribution function. Hence, the subintervals $(-\infty, q - 8]$ and $[q - 8, q]$ were considered. For the first subinterval, the following transformation of variable was used $y = b + (1 + t)/(t - 1)$, resulting in an integration interval given by $[-1, 1]$, as required to apply the Gauss-Legendre quadrature. With the same purpose, the transformation $z = (a - b)t/2 + (a + b)/2$ was applied to the second subinterval. The algorithm was denoted by dNMR. The approximation of (4) was computed by

$$\begin{aligned} f_Q(q) &= 2n(n-1) \left[\int_{-\infty}^{q-8} \phi(y)\phi(2q-y)[\Phi(2q-y) - \Phi(y)]^{n-2} dy \right. \\ &\quad \left. + \int_{q-8}^q \phi(y)\phi(2q-y)[\Phi(2q-y) - \Phi(y)]^{n-2} dy \right] \\ &\approx 2n(n-1) \left[\sum_{i=1}^m \frac{2}{(x_i-1)^2} \phi(y_i)\phi(2q-y_i)[\Phi(2q-y_i) - \Phi(y_i)]^{n-2} \right. \\ &\quad \left. + 4 \sum_{i=1}^m \phi(z_i)\phi(2q-z_i)[\Phi(2q-z_i) - \Phi(z_i)]^{n-2} \right] \tag{17} \end{aligned}$$

The choice of the value 8 was motivated by the same choice made by [Copenhaver and Holland \(1988\)](#) to the normal range distribution. Since the midrange distribution is more peaked than the range distribution, this value seems to be a good choice for the standardized normal midrange distribution. Further investigations could be undertaken to optimize the choice of these subintervals.

The pseudocode to compute the standardized normal midrange probability density function is

given by:

1. Input $q, n > 1, m \geq 2$;
2. Compute m nodes (x_i) and weights (w_i) , with $i = 1, 2, \dots, m$;
3. Transform x_i , from $[-1, 1]$ to $(-\infty, q - 8]$, using:
 $y_i = (q - 8) + (1 - x_i)/(x_i - 1)$, for $i = 1, 2, \dots, m$;
4. Compute $aux_i = \Phi(2q - y_i) - \Phi(y_i)$, for $i = 1, 2, \dots, m$;
5. Compute $aux1_i = \phi(y_i)$ and $aux2_i = \phi(2q - y_i)$, for $i = 1, 2, \dots, m$;
6. If $aux_i \leq 0$, then $aux_i = \epsilon$, for $i = 1, 2, \dots, m$;
7. Compute $fy_i = \ln(aux1_i) + (n - 1) \ln(aux_i) + \ln(aux2_i) + \ln(2) + \ln(n) + \ln(n - 1)$,
for $i = 1, 2, \dots, m$;
8. Apply $fy_i = \exp(fy_i)$, for $i = 1, 2, \dots, m$;
9. Compute $fy_i = [2/(x_i - 1)^2]fy_i$, for $i = 1, 2, \dots, m$;
10. Compute $I = \sum_{i=1}^m w_i fy_i$;
11. Transform x_i from $[-1, 1]$ to $[q - 8, q]$, using, $a = q - 8$ and $b = q$, where
 $y_i = [(b - a)/2]x_i + (a + b)/2$, for $i = 1, 2, \dots, m$;
12. Repeat steps 4 to 8, with the y_i from 11, for $i = 1, 2, \dots, m$;
13. Compute $fy_i = [(b - a)/2]fy_i$, for $i = 1, 2, \dots, m$;
14. Compute $I = I + \sum_{i=1}^m w_i fy_i$;
15. Return I .

The standardized normal midrange cumulative distribution function

The algorithm for computing the cumulative distribution function of standardized normal midrange, expression (5), was developed using the Gauss-Legendre quadrature. This will be essential in the construction of the externally studentized normal midrange. First, the integration interval was divided into two subintervals to achieve higher accuracy as suggested by Quarteroni et al. (2007) and the following division was considered:

$$F_Q(q) = \int_{-\infty}^{q-8} n\phi(y)[\Phi(2q - y) - \Phi(y)]^{n-1} dy + \int_{q-8}^q n\phi(y)[\Phi(2q - y) - \Phi(y)]^{n-1} dy. \quad (18)$$

In these two parts, different variable transformations were considered. In the first part, the transformation $y = (q - 8) + (1 + t)/(t - 1)$ was applied to obtain an integration interval $[-1, 1]$, needed to solve this integral by Gauss-Legendre quadrature. Considering y_i obtained by the above transformation, where $t = x_i$ is the i th Gauss-Legendre quadrature node, $i = 1, 2, \dots, m$, and m is the number of quadrature points, then the first part of the integral (18) was approximated by using the transformations (11) and (14). Hence,

$$I_1(q) = \int_{-\infty}^{q-8} n\phi(y)[\Phi(2q - y) - \Phi(y)]^{n-1} dy \approx n \sum_{i=1}^m w_i \frac{2}{(x_i - 1)^2} \phi(x_i) [\Phi(2q - y_i) - \Phi(y_i)]^{n-1}. \quad (19)$$

For the second part, the transformation $y = (b - a)t/2 + (a + b)/2$ was used, with the same goal presented for the first transformation. These changes were implemented implicitly as a numerical device, as shown above. Therefore, the second part of the integral (18) was approximated by using the transformations (12) and (15). It was given by

$$I_2(q) = \int_{q-8}^q n\phi(y)[\Phi(2q - y) - \Phi(y)]^{n-1} dy \approx \frac{n(b - a)}{2} \sum_{i=1}^m w_i \phi(y_i) [\Phi(2q - y_i) - \Phi(y_i)]^{n-1}, \quad (20)$$

where $b = q$ and $a = q - 8$.

Considering the expressions (19) and (20) the cumulative distribution function (18) was approximated by

$$F_Q(q) \approx I_1(q) + I_2(q), \quad (21)$$

in the same way as the numerical approximation adopted by McBane (2006) and by Verma and Suarez (2014) for obtaining the probability density and cumulative distribution functions and critical values for the range ratio statistics. The main difference between those works and the present results is the use of the Gauss-Legendre quadrature, besides the distribution functions where the quadrature was applied. All other integrals computed in this work will be based on this type of numerical approximation.

In the algorithm pseudocode, the standardized normal midrange denoted by w , is replaced by q , following the notation of expression (18), avoiding confusion with the quadratures weights. Hence, the pseudocode using the Gauss-Legendre quadrature to obtain the cumulative distribution function of the standardized normal midrange (21), which is denoted by pNMR, is given by:

1. Input $n > 1$, q , and $m \geq 2$;
2. Compute m nodes (x_i) and weights (w_i), with $i = 1, 2, \dots, m$;
3. Transform x_i , from interval $[-1, 1]$ to $(-\infty, q - 8]$: $y_i = (q - 8) + (1 + x_i)/(x_i - 1)$;
4. Compute $aux1_i = \Phi(y_i)$;
5. Compute $aux2_i = \phi(y_i)$;
6. Compute $aux_i = \Phi(2q - y_i) - aux1_i$;
7. If $aux_i \leq 0$, then $aux_i = \epsilon$;
8. Compute $fy_i = \log(aux2_i) + (n - 1) \log(aux_i)$;
9. Compute $fy_i = \exp(fy_i)$;
10. Compute $fy_i = n(2/(x_i - 1)^2)fy_i$;
11. The quantities in steps 3-10, should be computed for each node x_i , $i = 1, 2, \dots, m$. Therefore, compute $I = \sum_{i=1}^m w_i fy_i$;
12. Set $a = q - 8$ and $b = q$;
13. Transform x_i from interval $[-1, 1]$ to $[q - 8, q]$: $y_i = [(b - a)/2]x_i + (a + b)/2$, for $i = 1, 2, \dots, m$;
14. Repeat steps 4 to 9, with y_i of step 13, for $i = 1, 2, \dots, m$;
15. Compute $fy_i = n(b - a)/2 \times fy_i$, for $i = 1, 2, \dots, m$;
16. Compute $I = I + \sum_{i=1}^m w_i fy_i$;
17. Return I .

Standardized normal midrange quantile function

The algorithm qNMR was derived to compute the standardized normal midrange quantile function. The pseudocode makes use of the Newton-Raphson, expression (16), and is given by:

1. Input $n > 1$, $m \geq 2$, and $0 < p < 1$;
2. Set $eps = 1 \times 10^{-13}$, for the error tolerance, and $maxIt = 5000$, for the maximum number of iterations;
3. Get initial estimate of q , called q_0 , by:
 3. a. If $p < 0.5$, then $q_0 = -0.5$, else go to step 3b;
 3. b. If $p > 0.5$, then $q_0 = 0.5$, else set $q_0 = 0$;
4. $it = 0$;
5. While ($it \leq maxIt$) do
 - $cdf = \text{pNMR}(q_0, n, np)$;
 - $pdf = \text{dNMR}(q_0, n, np)$;
 - $q_1 = q_0 - (cdf - p)/pdf$;
 - go to step 6;
6. If $|q_1 - q_0| \leq eps$, then return q_1 and exit; otherwise, go to step 5, but first update the interactions counter $it = it + 1$ of Newton-Raphson method and do $q_0 = q_1$. In step 5, if this counter exceeds the limit $maxIt$, go to step 7;
7. print the error message: "iterative process did not achieve convergence in $maxIt$ steps."

Externally studentized normal midrange probability density function

For computing the externally studentized normal midrange probability density function, given in (6), note that the innermost integral is the probability density function of the standardized normal midrange, given by (4) and (17). Thus, the dNMR algorithm presented above was reused for computing the probability density function of interest. Also, the variable x (standardized standard deviation) is replaced by s in the implemented algorithm, avoiding confusion with the quadrature nodes. The outermost integral was divided into two parts and a Gauss-Legendre quadrature was also applied in each part. The integration intervals are given by $[0, 1]$ and $[1, \infty)$ for the same reasons mentioned above. Thus,

$$\begin{aligned} f(q; n, \nu) &= \int_0^\infty f_Q(sq)f(s; \nu)ds \\ &= \int_0^1 f_Q(sq)f(s; \nu)ds + \int_1^\infty f_Q(sq)f(s; \nu)ds, \end{aligned} \tag{22}$$

where $f(sq)$ was approximated by expression (17), with q substituted by sq , and computed by the algorithm dNMR, described in the previous section.

The innermost integral multiplied by the probability density function of the variable s , $f(s; \nu)$, was computed by

$$d(q; s, n, \nu) = \text{dNMR}(sq, n, m) \times \frac{\nu^{\nu/2}}{\Gamma(\nu/2)2^{\nu/2-1}} s^{\nu-1} e^{-\nu s^2/2} \tag{23}$$

The auxiliary algorithm dSMR_aux was constructed to compute (23). Its pseudocode is given by:

1. Input s (quadrature node at the i th step), $q, n > 1$ and $m \geq 2$;
2. $Y = \text{dNMR}(s \times q, n, m)$;
3. Compute

$$dsln = (\nu n/2) \ln(\nu n) - \ln \Gamma(\nu n/2) - (\nu n/2 - 1) \ln(2) + (\nu n - 1) \ln(s) - \nu n \times s^2/2;$$

4. Compute $fx = s \times \exp(dsln)$;
5. Compute $d = fx \times Y$;
6. Return d .

Each integration subinterval of (22) was appropriately transformed enabling the Gauss-Legendre quadrature to be applied for approximating the integral (6). For this purpose, in the first integration subinterval, $[0, 1]$, the transformation $y = (a - b)t/2 + (a + b)/2$ was adopted. Hence,

$$\begin{aligned} I_1(q; n, \nu) &= \int_0^1 f_Q(sq)f(s; \nu)ds \\ &\approx \sum_{i=1}^m w_i \frac{1}{2} d(q; y_i, n, \nu), \end{aligned} \tag{24}$$

where $d(q; y_i, n, \nu)$ is computed by (23), with corresponding pseudocode given by dSMR_aux.

In the second subinterval of $[1, \infty)$, the transformation (13), with $a = 1$, was applied, being given by $y = 1 + (1 + t)/(1 - t)$. Thus, considering the quadrature nodes x_i 's, the integral was approximated by

$$\begin{aligned} I_2(q; n, \nu) &= \int_1^\infty f_Q(sq)f(s; \nu)ds \\ &\approx \sum_{i=1}^m w_i \frac{2}{(1 - x_i)^2} d(q; y_i, n, \nu). \end{aligned} \tag{25}$$

The integration (6) was approximated by

$$f(q; n, \nu) \approx I_1(q; n, \nu) + I_2(q; n, \nu), \tag{26}$$

where $I_1(q; n, \nu)$ and $I_2(q; n, \nu)$ were computed by (24) and (25), respectively.

Again, all transformations were implicitly applied as a numerical integration device only. The expression (26) was used to obtain the pseudocode for the computation of the probability density function of the externally studentized normal midrange, denoted dMR. It is given by:

1. Input $q, n > 1, nu > 0$ and $m \geq 2$;
2. Compute m nodes (x_i) and weights (w_i), with $i = 1, 2, \dots, m$;
3. Transform the variable x_i from $[-1, 1]$ to $[0, 1]$, using:
 $y_i = 0,5x_i + 0,5$, for $i = 1, 2, \dots, m$;
4. Compute $fy_i = \frac{1}{2}dSMR_aux(y_i, q, n, m)$, for $i = 1, 2, \dots, m$;
5. Compute $I = \sum_{i=1}^m w_i fy_i$;
6. Transform x_i from $[-1, 1]$ to $[1, \infty]$:
 $y_i = 1 + (1 + x_i)/(1 - x_i)$, for $i = 1, 2, \dots, m$;
7. Compute $fy_i = dSMR_aux(y_i, q, n, m)$, for $i = 1, 2, \dots, m$;
8. Compute $fy_i = \ln(fy_i) + \ln(2) - 2 \ln(1 - x_i)$, for $i = 1, 2, \dots, m$;
9. Compute $fy_i = \exp(fy_i)$, for $i = 1, 2, \dots, m$;
10. Compute $I = I + \sum_{i=1}^m (fy_i \times w_i)$;
11. Return I .

Quadratures of 64 points are sufficient for most circumstances, although it is possible to increase the number of quadrature points and achieve the desired accuracy. The Newton-Raphson method was used to compute quantiles, since the probability density function, the first derivative, was available. The following subsections describe the algorithms and show their pseudocodes.

The externally studentized normal midrange cumulative distribution function

Note that the innermost integral of the externally studentized normal midrange cumulative distribution function, given in (7), is the cumulative distribution function of standardized normal midrange, given by (5) and (18). Thus, the pNMR algorithm presented above was reused for computing the cumulative distribution function of interest. Also, the variable x (standardized standard deviation) is replaced by s in the implemented algorithm, avoiding confusion with the quadrature nodes. The outermost integral was divided into two parts and a Gauss-Legendre quadrature was also applied in each part. The integration intervals are given by $[0, 1]$ and $[1, \infty)$. The limit 1 was chosen since it is the modal value of the standardized normal standard deviation probability density function $f(x; \nu)$, given in expression (3), following the ideas of [Copenhaver and Holland \(1988\)](#). Thus

$$\begin{aligned} F(q; n, \nu) &= \int_0^{\infty} F_Q(sq) f(s; \nu) ds \\ &= \int_0^1 F_Q(sq) f(s; \nu) ds + \int_1^{\infty} F_Q(sq) f(s; \nu) ds, \end{aligned} \quad (27)$$

where $F_Q(sq)$ was given by expression (18), with q substituted by sq , and computed by the algorithm pNMR, described in the previous section.

The innermost integral multiplied by the probability density function of the variable s , $f(s; \nu)$, was computed by

$$I(q; s, n, \nu) = \text{pNMR}(sq, n, m) \times \frac{\nu^{\nu/2}}{\Gamma(\nu/2) 2^{\nu/2-1}} s^{\nu-1} e^{-\nu s^2/2} \quad (28)$$

Therefore, an auxiliary algorithm, denoted by pNMR_aux, was constructed to compute (28). The pseudocode is given by:

1. Input $q, s, n > 1, nu > 0$ and $m \geq 2$;
2. $I = \text{pNMR}(s \times q, n, m)$;
3. Compute
 $fx = (nu/2) \ln(nu) - \ln \Gamma(nu/2) - (nu/2 - 1) \ln(2) + (nu - 1) \ln(s) - s \times nu \times s/2$;
4. Apply $fx = \exp(fx)$;
5. Compute $I = fx \times I$;
6. Return I .

Each subinterval of (27) was appropriately transformed enabling the Gauss-Legendre quadrature to be applied for solving the integral (7). For this purpose, in the first integration subinterval, $[0, 1]$, the transformation $y = (a - b)t/2 + (a + b)/2$, given by (12) and (15), was adopted. Hence, considering all quadrature nodes x_i 's, it follows that

$$\begin{aligned} I_1(q; n, \nu) &= \int_0^1 F_Q(sq) f(s; \nu) ds \\ &\approx \sum_{i=1}^m w_i \frac{1}{2} I(q; y_i, n, \nu), \end{aligned} \quad (29)$$

where $I(q; y_i, n, \nu)$ is given by (28).

In the second subinterval of $[1, \infty)$, the transformations (10) and (13), with $a = 1$, was applied and it is given by $y = 1 + (1 + t)/(1 - t)$. Thus, considering the quadrature nodes x_i 's, the integral was approximated by

$$\begin{aligned} I_2(q; n, \nu) &= \int_1^\infty F_Q(sq) f(s; \nu) ds \\ &\approx \sum_{i=1}^m w_i \frac{2}{(1 - x_i)^2} I(q; y_i, n, \nu). \end{aligned} \quad (30)$$

The integration (7) was computed by

$$F(q; n, \nu) \approx I_1(q; n, \nu) + I_2(q; n, \nu), \quad (31)$$

using the results of (29) and (30).

All transformations were implicitly applied, i.e., they were only a numerical integration device. The pseudocode for the computation of the cumulative distribution function of the externally studentized normal midrange, denoted pMR, applies the ideas of expression (31) and is given by:

1. Input $q, nu > 0, n > 1$, and $m \geq 2$;
2. Compute m nodes (x_i) and weights (w_i), with $i = 1, 2, \dots, m$;
3. For the first part of the split integral, transform: $y_i = 0.5x_i + 0.5$, for $i = 1, 2, \dots, m$;
4. Compute $fy_i = \frac{1}{2} \text{pNMR_aux}(q, y_i, n, m)$, for $i = 1, 2, \dots, m$;
5. Compute $I = \sum_{i=1}^m w_i fy_i$;
6. For the second part of the split integral, transform: $y_i = 1 + (1 + x_i)/(1 - x_i)$, for $i = 1, 2, \dots, m$;
7. Compute $fy_i = \text{pNMR_aux}(q, y_i, n, m)$, for $i = 1, 2, \dots, m$;
8. Compute $fy_i = \ln(fy_i) + \ln(2) - 2 \ln(1 - x_i)$, for $i = 1, 2, \dots, m$;
9. Hence, compute $fy_i = \exp(fy_i)$, for $i = 1, 2, \dots, m$;
10. Compute $I = I + \sum_{i=1}^m w_i fy_i$;
11. Return I .

Externally studentized normal midrange quantile function

For the externally studentized normal midrange quantile function, the qMR algorithm was constructed. The algorithm applies the Newton-Raphson method and depends on the pMR and dMR methods. Its pseudocode is:

1. Input: $n > 1, m \geq 2$, and $0 < p < 1$;
2. Set $eps = 1 \times 10^{-13}$, for the error tolerance, and $maxIt = 5000$, for the maximum number of iterations;
3. Get initial estimate of q , called q_0 , by:
 - 3.a. If $p < 0.5$, then $q_0 = -0.5$, else go to step 3b;
 - 3.b. If $p > 0.5$, then $q_0 = 0.5$, else set $q_0 = 0$;
4. $it = 0$;

5. While ($it \leq maxIt$) do
 - $cdf = \text{pMR}(q_0, n, np)$;
 - $pdf = \text{dMR}(q_0, n, np)$;
 - $q_1 = q_0 - (cdf - p)/pdf$;
 - go to step 6;
6. If $|q_1 - q_0| \leq eps$, then return q_1 and exit; otherwise, go to step 5 but first update the interactions counter $it = it + 1$ of Newton-Raphson method and do $q_0 = q_1$. In step 5, if this counter exceeds the limit $maxIt$, go to step 7;
7. print the error message: "iterative process did not achieve convergence in $maxIt$ steps."

The random number generator

To generate random sample of size N from the normal midrange distribution, the algorithm rMR was constructed, with parameter n and ν . First, a sample from the standard normal distribution of size n , X_1, X_2, \dots, X_n , was simulated, where $X_i \stackrel{\text{iid}}{\sim} N(0, 1)$. A chi-square variable $U \sim \chi_\nu^2$, with degrees of freedom ν , was also simulated, and the following transformation was computed:

$$Q = \frac{[\max(X_i) + \min(X_i)]/2}{\sqrt{\frac{U}{\nu}}}.$$

With infinite degrees of freedom, then the following transformation was considered instead

$$W = \frac{\max(X_i) + \min(X_i)}{2}.$$

The process was repeated N times and the values of Q or W , depending on the considered case, are stored in a vector of size N .

This distribution can be used to obtain cumulative probabilities and quantiles. Thus, for example, given a quantile q or w , the cumulative probability function, ensuring that N is large, is computed in an approximated way by:

$$P(Q \leq q) = \frac{\sum_{i=1}^N I(Q_i \leq q)}{N}, \quad \nu < \infty$$

$$P(W \leq w) = \frac{\sum_{i=1}^N I(W_i \leq w)}{N}, \quad \nu = \infty.$$

The random number generator rMR returns the vector of N realizations of Q or W and its pseudocode is presented below. The rMR generator is dependent of the R functions rnorm() and rchisq(), for obtaining a random vector of standard normal independently variables and a chi-square realization, respectively. The matrix(x, p, q) function, presented below, creates a matrix of dimension $p \times q$ filled by columns with the elements given by the vector x .

1. Input: $N > 1, n > 1$ and $\nu > 0$;
2. $x = \text{rnorm}(N \times n)$
3. $X = \text{matrix}(x, N, n)$ of dimension $(N \times n)$;
4. if $nu = \infty$, then go to (7), else go to (5);
5. $u = \text{rchisq}(N, \nu)$;
6. $X = X / \sqrt{(u/\nu)}$;
7. for $i = 1$ to N do:
 - $\text{rMR}[i] = [\max(X[i,]) + \min(X[i,])]/2$;
8. return rMR.

Details of the SMR package

The package SMR provides the following functions, where np is the number of nodes and weights of the Gauss-Legendre quadrature:

- `dSMR(x, size, df, np=32)`: computes values of the probability density function, given in (6) or (4);
- `pSMR(q, size, df, np=32)`: computes values of the cumulative distribution function, given in (7) or (5);
- `qSMR(p, size, df, np=32)`: computes quantiles of the externally studentized normal midrange;
- `rSMR(n, size, df=Inf)`: drawn a random sample of size n from the externally studentized normal midrange.

The value of the argument `df` can be finite or infinity. If `df=Inf`, values of the probability density, cumulative distribution and quantile functions of the normal midrange (standardized normal midrange) are computed. If the argument `df` is not specified in the `rSMR` function, the default value `Inf` is used and random samples from the normal midrange distribution are drawn. The other functions presented earlier in the previous section are internal algorithms of the **SMR** package.

As an illustration, consider the following examples:

```
library(SMR)
set.seed(10701)
q <- 2 # quantile
x <- 2 # quantile
p <- 0.9 # probability
n <- 10 # sample size to be simulated
size <- 5 # normal sample size
df <- 3 # degrees of freedom
np <- 32 # number of points of the Gaussian quadrature

dSMR(x, size, df, np) # SMR pdf
[1] 0.01926172

pSMR(q, size, df, np) # SMR cdf
[1] 0.9851739

qSMR(p, size, df, np) # SMR quantile
[1] 0.8350065

rSMR(n, size, df) # random sample of the SMR distribution
[1] 0.35108979 0.33786356 -0.13753510 -0.58741681 -0.40358907
[6] -0.72528615 0.45845331 0.08906021 -1.64157684 0.07022362
```

In the case $\nu = \infty$, the result of the standardized normal midrange distribution is returned, as illustrated by the following example:

```
library(SMR)
q <- 2 # quantile
x <- 2 # quantile
p <- 0.9 # cumulative probability
n <- 10 # sample size to be simulated
size <- 5 # normal sample size
df <- Inf # degrees of freedom
np <- 32 # number of points of the Gaussian quadrature

dSMR(x, size, df, np) # normal MR pdf
[1] 0.0004487675

pSMR(q, size, df, np) # normal MR cdf
[1] 0.9999408

qSMR(p, size, df, np) # normal MR quantile
[1] 0.6531507

rSMR(n, size, df,) # random sample of the normal MR distribution
[1] -0.52475079 0.10198842 -0.38647236 0.18939367 0.17756023
[6] -1.03384242 0.35608349 1.00629514 0.06360581 0.70835452
```

A concrete application of the **SMR** package on real dataset is now considered. We use the data on nitrogen contents of red clover plants presented in [Steel and Torrie \(1980\)](#), page 180. A completely randomized design with one factor with 6 levels and $r = 5$ replications were carried out. The mean squared error (MSE) was 11.7887, with $\nu = 24$ degrees of freedom. The factor means is presented in Table 1. A multiple comparison procedure was suggested by [Batista \(2012\)](#). The suggested procedure was similar to the Tukey test ([Tukey, 1953](#)). First the least significant difference of the proposed test is computed by

$$\Delta = 2q_{1-\alpha/2;n,\nu} \sqrt{\frac{MSE}{r}},$$

where $q_{1-\alpha/2;n,\nu}$ is the $100\alpha/2\%$ quantile of the externally studentized normal midrange. This value can be computed by using the function `qSMR(0.975, 6, 24, np=32)=1.0049` of the **SMR** package, for the significance level $\alpha = 0.05$.

The Δ value in this case is

$$\Delta = 2 \times 1.0049 \times \sqrt{\frac{11.7887}{5}} = 3.0859.$$

Finally, the test is performed in the same way of the Tukey test. The results of the proposed midrange and Tukey tests are shown in Table 1. The proposed midrange test show no ambiguous results in this example, as happens with the Tukey test (two or more letters per level).

Table 1: The midrange and Tukey multiple comparison procedures applied to the nitrogen contents of 6 red clover plants presented in [Steel and Torrie \(1980\)](#), page 180.

Levels	means	Tukey*	Midrange*
5	13.26	c	d
3	14.64	c	d
6	18.70	cb	c
4	19.92	cb	c
2	23.98	ba	b
1	28.82	a	a

* Means with the same letter are not significantly different.

Concluding remarks

The importance of the externally studentized normal midrange distribution can be enormous in the analysis of experiments, since the midrange estimator is more efficient than the sample mean in platykurtic distributions. Therefore, this distribution could be useful in the proposition of multiple comparison procedures, that could potentially show better results (more robust and powerful) than the traditional tests based on the externally studentized normal range.

This package is easy to use and shows very high accuracy. The accuracy of critical values is estimated through the computation of the difference between two results using different numbers of quadrature points according to the expression (9). The number of quadrature points can be chosen and for most cases, 32 points are enough to achieve high accuracy. Monte Carlo simulations were also used to evaluate the accuracy of the proposed methods and the consistency of the **SMR** package. The algorithms were proposed and implemented using Gauss-Legendre quadrature and the Newton-Raphson method in R software, resulting in the **SMR** package, that is available for download from CRAN. An important aspect is that the **SMR** package is written in R. Code in Fortran or C can make the **SMR** functions faster, a feature that is planned for future releases of the package. Another important aspect is the possibility of using in future releases two-dimensional interpolations when `dSMR`, `pSMR` and `qSMR` are called with long vectors of arguments x , q , and p . In this case, two-dimensional grids for $\{x, q, \text{ and } p\}$ and $\{size\}$ can be computed offline and the `approx` function used to interpolate.

Acknowledgments

We would like to thank CNPq and CAPES for their financial support. We would also like to thank two anonymous referees and the Editor for helpful comments which have resulted in an improved manuscript and package.

Bibliography

- B. D. O. Batista. Externally studentized normal midrange exact distribution and development of an R package using gaussian quadrature. Master's thesis, Federal university of Lavras, Brazil, 2012. [p124, 134]
- B. D. O. Batista and D. F. Ferreira. *SMR: Externally Studentized Midrange Distribution*, 2014. URL <http://CRAN.R-project.org/package=SMR>. R package version 2.0.1. [p123, 124]
- T. S. Chihara. *An introduction to Orthogonal Polynomials*, volume 13 of *Mathematics and its Applications Series*. Gordon and Breach, New York, 1978. [p125]
- M. D. Copenhaver and B. Holland. Computation of the distribution of the maximum studentized range statistic with application to multiple significance testing of simple effects. *Journal of Statistical Computation and Simulation*, 30(1):1–15, 1988. [p123, 126, 130]
- H. A. David and H. N. Nagaraja. *Order Statistics*. John Wiley & Sons, Canada, 2003. [p124]
- P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, New York, 2 edition, 1984. [p125]
- A. Gil, J. Segura, and N. Temme. *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, 2007. [p125]
- E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, New York, 1958. [p124]
- F. B. Hildebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 2 edition, 1974. [p124, 125]
- R. E. Lund and J. R. Lund. Algorithm AS 190: Probabilities and upper quantiles for the studentized range. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 32(2):pp. 204–210, 1983. [p123]
- G. C. McBane. Programs to compute distribution functions and critical values for extreme value ratios for outlier detection. *Journal of Statistical Software*, 16(3):1–9, 5 2006. [p128]
- G. Miller. *Numerical Analysis for Engineers and Scientists*. Cambridge University Press, New York, 2014. [p126]
- D. Newman. The distribution of range in samples from a normal population, expressed in terms of an independent estimate of standard deviation. *Biometrika*, 31(1/2):pp. 20–30, 1939. [p124]
- F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, editors. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010. [p126]
- K. C. S. Pillai. On the distribution of midrange and semi-range in samples from a normal population. *Annals of Mathematical Statistics*, 21:100–105, 1950. [p124]
- A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Texts in Applied Mathematics. Springer, 2007. [p127]
- P. R. Rider. The midrange of a sample as an estimator of the population midrange. *J. Amer. Statist. Ass.*, 52(280):537–542, Dec. 1957. [p123]
- S. R. Searle. *Linear models for unbalanced data*. J. wiley, New York, 1987. [p124]
- R. Steel and J. Torrie. *Principles and Procedures of Statistics: A Biometrical Approach*. McGraw-Hill series in probability and statistics. McGraw-Hill, New York, 2 edition, 1980. [p134]
- J. W. Tukey. The problem of multiple comparisons. *Unpublished Dittoed Notes, Princeton University*, 1953. [p134]
- M. Verma and M. C. Suarez. Dixontest.criticalvalues: A computer code to calculate critical values for the dixon statistical data treatment approach. *Journal of Statistical Software*, 57(2):1–12, 3 2014. [p128]

Ben Dêivide de Oliveira Batista
Department of Exact Sciences, Federal University of Lavras
37200-000, Lavras, Brazil
URL: <http://ufla.br>
ben.deivide@gmail.com

Daniel Furtado Ferreira
Department of Exact Sciences, Federal University of Lavras
37200-000, Lavras, Brazil
URL: <http://danielfurtado.url.ph>
danielff@dex.ufla.br

Farewell's Linear Increments Model for Missing Data: The FLIM Package

by Rune Hoff, Jon Michael Gran and Daniel Farewell

Abstract Missing data is common in longitudinal studies. We present a package for Farewell's Linear Increments Model for Missing Data (the **FLIM** package), which can be used to fit linear models for observed increments of longitudinal processes and impute missing data. The method is valid for data with regular observation patterns. The end result is a list of fitted models and a hypothetical complete dataset corresponding to the data we might have observed had individuals not been missing. The **FLIM** package may also be applied to longitudinal studies for causal analysis, by considering counterfactual data as missing data - for instance to compare the effect of different treatments when only data from observational studies are available. The aim of this article is to give an introduction to the **FLIM** package and to demonstrate how the package can be applied.

Introduction

Longitudinal data consist of repeated measurements recorded for a group of individuals over a given time period. The resultant datasets are often incomplete, i.e. missing responses for some individuals. Incomplete datasets are more troublesome to analyse, since one has to assess the mechanism behind the missing data. If a response measure itself is linked to the probability of being missing, a standard analysis of the observed data may be severely biased.

Data from repeated observations have a natural time order. Standard likelihood-based models for quantitative analysis usually ignore time order and treat longitudinal data as cluster data (Hogan et al., 2004), giving little regard to where missingness takes place or to how the response processes evolve. Farewell (2006) and Diggle et al. (2007) introduced the linear increments model as a tool for dealing with missing data due to drop-out in a dynamic manner, that is, explicitly considering the time order of the responses and where drop-out occurred. The key concept in the linear increments model is the increments, which represent changes over time, and hence are representative of evolving response processes (Aalen and Gunnes, 2010).

As with any method that considers missing data, the linear increments model is not exempt from untestable assumptions. In the linear increments model, the discrete time-independent censoring (DTIC) assumption is adopted. This assumption states that the local characteristics of the response are unchanged by additional information about who is censored, or knowledge of who will, or will not, be censored at the next time point (Diggle et al., 2007). The DTIC assumption is similar to the sequential version of missing at random assumption (Hogan et al., 2004), and to independent censoring in survival analysis.

The **FLIM** package implements an autoregressive multivariate version of the linear increments model described by Aalen and Gunnes (2010). The end result is a hypothetical complete dataset that might have been observed had individuals not been missing, referred to as a hypothetical complete dataset, which can be used for further analysis. Because of the nature of the method, it is required that the longitudinal data exhibits a regular observation pattern with the same potential observation times for all individuals. The **FLIM** package may also be applied to longitudinal studies for causal analysis, by considering the counterfactuals as missing data. For instance, we may artificially censor individuals after they initiate treatment and apply the linear increments model to estimate counterfactual responses had treatment not been initiated.

Early work on the linear increments model focused on handling monotone missingness (i.e. when individuals are missing and do not return to the study; typically referred to as drop-out). The approach of Aalen and Gunnes (2010), which we adopt, includes both monotone and nonmonotone missingness (i.e. when individuals are missing for a certain amount of time, but later return to the study), and the required assumptions for both types of missingness are discussed.

The linear increments model

The linear increments model assumes a hypothetical complete dataset. The focus lies in the response that would have been observed if, contrary to fact, individuals did not drop out. Thus, the linear increments model makes explicit the possibility that dropping out can influence the response measurements, rather than just lead to missing data.

The model is, as the name implies, based on linear models for the increments, which represent

changes over time and thus capture the evolving response process (Aalen and Gunnes, 2010). At every time point, a linear model is fitted for each longitudinal response by regressing the observed increments onto lagged values of the response variables and possibly other covariates. Afterwards, missing data are imputed by estimating increments for the unobserved individuals and then adding these to the lagged values of their response variables.

To fix some notation, we start by describing the hypothetical complete dataset. Let $\tilde{Y}(t)$ be an $n \times m$ matrix of multivariate individual responses defined for a set of times $t \in \{0, \dots, k\}$, with $\tilde{Y}(0) = y_0$ being a matrix of fixed starting values for the processes. Further we write $X(t)$ for an $n \times p$ matrix of multivariate individual covariates that can contain both baseline and time dependent variables – note that any time dependent covariate is assumed to take on the same value as the last observed value on missing observations. Aalen and Gunnes (2010) define the increment of a response measure as $\Delta\tilde{Y}(t) = \tilde{Y}(t) - \tilde{Y}(t - 1)$, and we assume for each t that $\Delta\tilde{Y}(t)$ satisfies the model

$$\Delta\tilde{Y}(t) = X(t - 1)\alpha(t) + \tilde{Y}(t - 1)\beta(t) + \tilde{\epsilon}(t). \tag{1}$$

Here $\beta(t)$ is an $m \times m$ matrix of parameters, $\alpha(t)$ is an $p \times m$ matrix of parameters and $\tilde{\epsilon}(t)$ is an $n \times m$ error matrix. The error is defined as a zero mean martingale (Diggle et al., 2007). It follows that

$$\mathbb{E}(\Delta\tilde{Y}(t)|\mathcal{F}_{t-1}) = X(t - 1)\alpha(t) + \tilde{Y}(t - 1)\beta(t),$$

where \mathcal{F}_t is the history of responses, and covariates up to and including time t .

To estimate the parameters in (1), a nonparametric model is assumed over time, so there is no connection between $\alpha(t_1)$ and $\alpha(t_2)$ for different times t_1 and t_2 and neither for $\beta(t_1)$ and $\beta(t_2)$. The parameter matrices $\alpha(t)$ and $\beta(t)$ can be estimated in an unbiased manner by ordinary least squares from the observed increments (Aalen and Gunnes, 2010).

In order to simplify the derivations, we shall for the rest of this section assume that there only are responses and no covariates. To write out the estimator for $\beta(t)$, we define $R(t)$ and $R_0(t)$ as $n \times n$ diagonal matrices with response indicators on the diagonal. The i th diagonal element of $R(t)$ equals 1 when the complete set of response increments $\Delta\tilde{Y}_i(t)$ is observed for individual i , and $R_0(t)$ is defined similarly for the response measures. We assume that observation of an increment $\Delta\tilde{Y}_i(t)$ implies that both $\tilde{Y}_i(t)$ and $\tilde{Y}_i(t - 1)$ were observed. Define $Y(t)$ to be the observed data, so the increments can be written as $\Delta Y(t) = Y(t) - Y(t - 1)$ when they are well defined. Thus it is not necessarily the case that the observed values $Y(t)$ equals the sum $\sum \Delta Y(t)$. If we now write $U = R(t)Y(t - 1)$, the least squares estimate of $\beta(t)$ is:

$$\hat{\beta}(t) = (U^T U)^{-1} U^T \Delta Y(t). \tag{2}$$

Once linear models for the observed increments are fitted, estimates of the increments for missing individuals may be calculated and data can be imputed. The idea is to use real responses where they are available and then to substitute missing increments by iteratively updated estimates. The imputation scheme for the predicted hypothetical complete data is as following:

$$\begin{aligned} \tilde{Y}^{est}(0) &= y_0 \\ \Delta\tilde{Y}^{est}(t) &= (1 - R_0(t))\tilde{Y}^{est}(t - 1)\hat{\beta}(t) + R_0(t)(Y(t) - \tilde{Y}^{est}(t - 1)), \quad t = 1, \dots, k, \\ \tilde{Y}^{est}(t) &= \tilde{Y}^{est}(t - 1) + \Delta\tilde{Y}^{est}(t), \quad t = 1, \dots, k. \end{aligned}$$

When data is observed, the value $\tilde{Y}^{est}(t)$ just equals $Y(t)$. When there is missing data, the increments are predicted according to the model.

When applying the linear increments model, some issues might need to be resolved for the imputation to work. To obtain reasonable estimates, a sufficient number of observed increments must be available at all times. Sufficient number is a vague concept, but towards the end of a study there are typically less individuals left. Therefore, it might sometimes be advisable to stop imputation before the last observation times. Also, if the matrix $U^T U$ in (2) is singular at any time, such that it cannot be inverted, the routine is halted. At this point, one option is to do a Ridge regression to find β -estimates.

The DTIC assumption requirement for the data is discussed below – if the required assumptions hold, hypothetical complete datasets constructed using the above procedures will have the correct mean structure. However, they will not have the correct random error variation (Aalen and Gunnes, 2010). To deal with this, bootstrap samples of hypothetical complete datasets can be produced. This should not be too time-consuming, as iterations are not needed to find estimates. The bootstrap is done so that the same number of individuals as in the original data are sampled with replacement from the study population. If an individual is sampled, his or her entire observation matrix should be included. Missing data in a bootstrap sample are then imputed in the same manner as the original dataset, and inference can be based on these bootstrap samples. In the package, we have included the

possibility to calculate empirical bootstrap estimates of the standard errors for the hypothetical mean responses, and also an option to draw point wise 95% confidence bands around the hypothetical mean response curves.

To perform model check for the increments models, one can use the standard diagnostics plots for linear regression fits. These plots should reveal any serious misspecification of the model, and users should check for the usual signs of model misspecification in linear models – these are briefly explained when demonstrating the package later. We have included in the package an option to easily view the diagnostics plots that are produced for "lm" objects in R on the different time points where linear models were fitted.

The discrete time-independent censoring assumption

The DTIC assumption is covered with greater detail in [Gunnes et al. \(2009\)](#). Despite the term censoring, the DTIC is actually an assumption about missingness. The terminology comes from the analogy of independent censoring in survival analysis.

To write out the assumption, let $\mathcal{R}(t)$ be the past history of $R(t)$, i.e the history of the censoring process up to time $t - 1$. The censoring is considered to be independent of the hypothetical response process \tilde{Y} if, and only if,

$$E(\Delta\tilde{Y}(t)|\mathcal{F}_{t-1}, \mathcal{R}(t)) = E(\Delta\tilde{Y}(t)|\mathcal{F}_{t-1}), \forall t. \quad (3)$$

Independent censoring says that the local characteristics of \tilde{Y} are unchanged by additional information about the censoring process. This assumption ensures that the observed increments remain representative of the original study sample ([Diggle et al., 2007](#)). The DTIC assumption guarantees that the observed data will satisfy the model specified in (1) ([Aalen and Gunnes, 2010](#)).

Monotone and nonmonotone missingness

Early works on the linear increments model concentrated on censored or monotone missing data. Although the imputation technique may be valid when there are nonmonotone missing data, assumptions can prove harder to justify, and it might therefore be necessary to remove data from individuals with nonmonotone missingness. In many cases, if data are missing at only a few time points it would be foolish to ignore subsequent data. For instance, if only one measurement is missing for reasons unrelated to the evolution of the response process, then later measurements for the same individual should not be ignored. On the other hand, individuals who are missing from the study for an extended period and then return might not be representative of the population of the missing ones. The problem is to decide whether the data from these individuals should be used, which goes beyond the DTIC assumption ([Aalen and Gunnes, 2010](#)).

To decide whether monotone or nonmonotone missing data can be used, one must determine whether the probabilities of going missing and returning to the study are related to the progress of the complete response $\tilde{Y}(t)$. There are four possible situations,

1. Going missing "is unrelated" to the previous response progress. Returning "does not depend" on the progress while missing.
2. Going missing "is related" to the previous response progress. Returning "does not depend" on the progress while missing.
3. Going missing "is unrelated" to the previous response progress. Returning "does depend" on the progress while missing.
4. Going missing "is related" to the previous response progress. Returning "does depend" on the progress while missing.

In order to consistently use nonmonotone missing data, the probability of returning to the study should not depend on the response progress while missing, as is the case in situations 1 and 2 above. In situations where the probability of returning does depend on the response process while missing, data from observation times after an individual returns may need to be removed from the analysis. For all four situations above, monotone missing data can be used, since no considerations to any developments during absence have to be made. For a more elaborate discussion on the matter, see [Aalen and Gunnes \(2010\)](#).

Causal analysis

Methods for dealing with missing data have much in common with methods for estimating causal effects ([Aalen and Gunnes, 2010](#)). A typical problem in longitudinal studies is the comparison of

treatment effects when treatment is confounded with the condition of the individual rather than being randomised. For instance, if treatment is initiated with increasing probability as individuals get sicker, a naive analysis could conclude that individuals receiving treatment are worse off than if left untreated. Here the linear increments model can be adopted to estimate the counterfactual condition progress had treatment not been initiated, which could shed light on the treatment effectiveness. To see how this can be achieved, note that, as reported by [Hernán et al. \(2006\)](#), treatment initiation or change in treatment may be seen as artificial censoring.

To demonstrate how the linear increments model can be used to assess causal effects, for example, assume a situation where treatment A and treatment B are to be compared in their effectiveness of improving the condition of the patients. Rather than randomising treatment, the decision regarding type of treatment is made based on the patients' conditions, typically quantified by some measured response variable. Patients are given either treatment A or treatment B, and those receiving treatment A are moved to treatment B if their condition worsens. Depending on the purpose of the analysis, we may explicitly censor response values for individuals who changed from treatment A to treatment B after their last observation under treatment A. The data would be reconstructed as if these patients never initiated treatment B, but rather continued treatment A and were "missing" from follow-up for the remainder of the study.

It is possible to use **FLIM** to impute counterfactual response developments as described above. To do this, the dataset should include a factor variable that indicates that an intervention has taken place, for instance some form of treatment. This variable should be coded so that the value is zero whenever a patient is not intervened on, and 1 from the time point when intervention first took place. Intervention has to be of such nature that once initiated, the patient continuous to undergo this intervention for the remainder of the study period or until being censored or having an event. An example is shown on a simulated dataset later in the article.

The FLIM package

Applications of the linear increments model in the **FLIM** package are done with the core function `flim`:

```
flim(formula, data, id, obstime, t.values = NULL, method = "locf", lambda = NULL,
     art.cens=NULL)
```

To use the function, the following *mandatory arguments* must be entered. Note that `id` and `obstime` should be entered with quotation marks.

- `formula`: an R formula on the form $\text{response} \sim \text{predictors}$. The package fits models for the increments, so that $\text{formula} = Y \sim Y + X$ specifies the model $E[Y(t+1) - Y(t)] = \beta_0(t) + \beta_1(t)Y(t) + \beta_2(t)X(t)$. For several responses and the same set of predictors, `cbind(Y1, Y2) ~ predictors` can be used. For full flexibility, a list of formulae can be supplied.
- `data`: a data frame with the longitudinal dataset. See separate section later for specification requirements.
- `id`: name of the variable in data with unique ids for each subject.
- `obstime`: name of the variable with observation times.

In addition, there are some *optional arguments* that should be specified accordingly when needed.

- `t.values`: a vector with time points at which models should be fitted and missing data imputed. Models are fitted from the first time point all the way through to the penultimate time point. While data are imputed from the second time point up to the last one. If nothing is specified, this argument will be set to the unique time points contained in the data.
- `method`: method for filling in response values between observations if there is nonmonotone missingness. The options are "locf", "approx", "recursive" and "recursiveX". Default is "locf". See details in a separate section about this argument.
- `lambda`: ridge parameter for doing ridge regression in the linear fits. Default is OLS.
- `art.cens`: used to specify a 0-1 factor variable that will be used for artificial censoring. Artificial censoring is performed to response values for all subjects after the first switch from 0 to 1. Missing values will then be imputed as if subjects never made the switch. This is intended for advanced users and does not come with readily available tools for investigating results. The original dataset with added columns for counterfactual response values is stored in the fitted object.

Below is an example of how arguments can be entered. Imagine a dataset called `mydata` consisting of a cohort of HIV patients similar to e.g. [Aalen et al. \(2012\)](#). The patients are observed regularly, where they have among other variables, measured `cd4` and `rna` levels. Suppose `ids` and observation times are stored in the columns `pID` and `Obs`, respectively, and we wish to impute missing values for the two responses, `cd4` and `rna`. We also want to include the covariate `age` as a predictor. Assume our goal is to fit models and impute data for times $t \in \{1, 2, 3, \dots, 20\}$ – individuals may be observed on any arbitrary set of times contained within t as long as they are fully observed on their first observation, that is, all responses and covariates are registered. Any potential nonmonotone missing data are to be filled in by last observation carried forward (LOCF). To achieve this, we enter the following:

```
flimobject <- flim(cbind(cd4, rna) ~ cd4 + rna + age,
                 mydata, "pID", "Obs", t.values = 1:20)
```

This example would be equivalent to specifying these two models:

$$\begin{aligned} E[cd4(t+1) - cd4(t)] &= \beta_0(t) + \beta_1(t)cd4(t) + \beta_2(t)rna(t) + \beta_3(t)age \\ E[rna(t+1) - rna(t)] &= \alpha_0(t) + \alpha_1(t)cd4(t) + \alpha_2(t)rna(t) + \alpha_3(t)age \end{aligned}$$

A short stepwise review of the programme flow is as follows. First data are sorted by `id` and increasing time. Then a check for whether all individuals have a complete set of responses on their first observation is performed. After this, the dataset is prepared for imputation so that each individual has rows corresponding to the same set of observation times. If there is nonmonotone missingness, this is handled according to the `method` argument, which is described in greater detail later in the article. Subsequently, increments are calculated, and separate regressions performed at each time point.

To combine the reconstruction driven approach where missing data are imputed, and the task of fitting models for the increments, the linear regressions are performed successively by increasing observation time. This means that missing data can routinely be inserted based on previous imputed data and the model for increments at the current time.

The end result from applying `flim` is an object of class `"flim"` in R – a list containing among others the value `dataset` with the reconstructed data (hypothetical complete dataset) and `fit`, which contains the fitted models. Methods and functions for the class `"flim"` are as follows:

- `print(x)`
- `summary(x)`
- `plot(x, ...)`, see separate section on how to plot a `"flim"` object for additional arguments.
- `flimMean(x, response, grouping=NULL)`, calculates mean responses.
- `flimList(x)`, to assess model fits.
- `flimboot(x, R, counter = F)`, bootstraps a `flim` object.

The arguments are

- `x`: an object of class `"flim"`.
- `response`: name of a response variable.
- `grouping`: optional factor variable used in the model.
- `R`: number of bootstrap samples.
- `counter`: logical. If `TRUE` displays a bootstrap sample counter. Works by default on Mac and Linux platforms.

The functions `flimList` and `flimboot` take a fitted `"flim"` object as main argument and create objects of class corresponding to the function name. Examples are shown in the application section later. `flimList` is used to assess the model fits and has three usages

- `print(flimList(flimobject))` prints the predictor coefficients for all increment models on every included time point.
- `summary(flimList(flimobject))` gives estimated standard errors, `t`-values and `p`-values for all predictor coefficients in the linear models at every time point.
- `plot(flimList(flimobject, response))` plots model diagnostics in a 2x2 grid for the linear model for the specified response. Plots are shown for the first time point, then the second and so on when pressing `<ENTER>`.

Bootstrapping a `flim` object can be done with `flimboot`, which creates a `"flimboot"` object. The class has two working methods. Suppose `fbo` is a `"flimboot"` object, then

- `flimSD(fbo, response, grouping = NULL)` calculates empirical standard errors of the hypothetical bootstrap mean responses. The argument `response` chooses the response variable, and `grouping` is an optional factor variable.
- `plot(fbo, response, grouping=NULL, ...)` plots the hypothetical reconstructed mean response from the original dataset together with bootstrapped point wise 95% confidence bands. Takes the same arguments as `flimSD` and in addition any argument that can be passed to `plot`. By default the hypothetical means are fully drawn and the confidence bands stipulated.

The dataset

A long format longitudinal dataset, in which each observation is stored as a separate row, is required. Further, the dataset must contain a column for `id`, which uniquely identifies every individual in the study, and a column for the observation times. For each individual, the data should only include rows corresponding to observation times where at least one of the (potentially many) response variables is observed, missing data are denoted `NA`. For the iterative imputation to work, all individuals need a complete set of responses at their first observation time. The `panss` dataset (Diggle, 1998) included in the package is an example of how the data should look. If data are in wide format, one option is to transform them into long format with the `reshape` function in R.

The method argument

Specifying this argument in the `flim` function determines how missing data before drop-out should be treated; after drop-out, the linear increments model is used regardless of choice of method. There are four options, `"locf"` (default), `"approx"`, `"recursive"` and `"recursiveX"`.

For the option `"locf"`, missing values when there are available observed values on a later time point, are imputed by inserting the last observed value. To implement LOCF, we use the function `na.locf` in the `zoo` package (Zeileis and Grothendieck, 2005).

The second option is `"approx"`, or the approximation method, which imputes the missing data between observations by linear approximation, calculated with `na.approx` in `zoo`. Conceptually this approach may be more appealing than LOCF. The approximation method performs a linear interpolation to impute missing data between two observed values. The pitfall of this approach is that it may go against our prime directive: to consider specifically the time order of the measurements, and to use information contained in the response histories available at the time when data is missing, to impute data. Indeed, the approximation method violates this by looking ahead at future response values, which is in some sense cheating.

The third option is `"recursive"`, which uses the linear increments model to fill in data for missing individuals throughout the entire dataset – regardless of whether they are observed again on a later time or not.

The final option is `"recursiveX"`, which is similar to `"recursive"`. Unlike `"recursive"`, when fitting models, this method utilises data from observations just as patients return to the study, where there are no observed increments since the values are missing until the patients return. This is achieved by using estimated increments based on previously imputed data.

Plotting a flim object

The `plot` function can be used to plot the hypothetical mean responses of the imputed data, the mean responses of the observed data and a spaghetti plot of all individual trajectories. The default method is to plot mean responses where both hypothetical and observed curves are shown.

```
plot(x, response, grouping = NULL, ylim = NULL, col = NULL, naive = T,
     lty = 1:2, ptype = "mean", ylab = "Response", xlab = "Times", ...)
```

Arguments for plotting a `flim` object are:

- `x`: the `"flim"` object.
- `response`: name of a longitudinal response.
- `grouping`: optional group/factor variable used in the model.
- `ptype`: decides plot type, default is `"mean"` for mean responses, while `ptype="spa"` will give spaghetti type plot.
- `naive`: logical. If `TRUE` the plot will contain the observed means as well.

The rest of the arguments are standard arguments to plot that have been given default values – these may be altered by the user. Note that `lty` is a vector of length two and specifies the linetype of the observed mean response and hypothetical mean response respectively. An example of plotting a "flim" object is shown in the application section.

Bootstrapping with FLIM

Once a "flim" object is created, it can be bootstrapped to create a list of "flim" objects, each using a new dataset that has been resampled from the original data. The resampling is done with replacements on the id numbers so that every time an individual is selected to the bootstrap data, their full observation cluster is included. Bootstrapping in the linear increments model allows us to easily calculate standard errors for the hypothetical mean response.

The bootstrap function is called `bootflim`. Because `flim` can handle a lot of different model specifications and imputes data iteratively, the routine may be slower than a script working for one specific model. Nevertheless, because no iterations are required to reach the estimates in the models, time is not a major issue – with our example data `panss`, 100 samples were fitted in around 10 seconds on a laptop.

After a "flim" object have been bootstrapped, the hypothetical mean using imputed data in the original dataset can easily be plotted with confidence bands as shown in the application section – these confidence bands are point-wise and 95%.

Application

In this section, we will analyse data from an investigation of the effect on PANSS score of different schizophrenia treatments. A longitudinal study was conducted to compare three treatment regimens, haloperidol, placebo and risperidone among patients with schizophrenia. Patients were randomised to treatments at time 0, and followed for 8 weeks where they had PANSS score evaluated. PANSS is a score measure that quantifies the severity of schizophrenic symptoms, and the goal of the study was to compare the effectiveness of treatments in improving (reducing) the mean PANSS score. The data consist of 685 observations among 150 patients, and were extracted from a larger, confidential dataset from a randomised clinical trial. An analysis of the complete dataset can be found in [Diggle \(1998\)](#).

```
> install.packages("FLIM")
> library(FLIM)
> data(panss)
> head(panss, 8)
  treat time  Y id
1     1    0  91  1
2     2    0  72  2
3     1    0 108  3
4     1    1 110  3
5     1    0 106  4
6     1    1  93  4
7     1    0  77  5
8     1    1  80  5
```

To fit the linear models for increments and reconstruct the data, one should use the main function `flim`. Here a model where the PANSS score increments are regressed onto the score value and the treatment group is fitted, which corresponds to entering `Y~Y+factor(treat)` in the function. The remaining mandatory arguments are filled in as needed, and the extra arguments are left unspecified.

```
panss.flim <- flim(Y~Y+factor(treat), data=panss, id="id", obstime="time")
```

The stored object `panss.flim` is now of class type "flim"; some additional information about the object, over the standard print, can be had by using `summary` on the object. If users want direct access to the reconstructed data, these are stored in the value `dataset`. The linear models fitted on each time point are stored as a list in the value `fit`.

Instead of directly accessing the data and the models, users may want to take advantage of the built-in functions available to analyse the results. We start by calculating the mean response with `flimMean`:

```
> flimMean(panss.flim, "Y")
  hypothetical observed
```

```

0 92.02000 92.02000
1 87.39672 87.44595
2 86.01734 83.07874
4 85.78134 80.25926
6 87.88868 77.42857
8 88.52504 76.00000
    
```

This is displaying the mean response of the variable Y for the observed data and the hypothetical reconstructed data.

In order to perform separate calculations for the different treatment groups, the grouping argument must be specified. The first three columns in the print below show the mean responses for the hypothetical reconstructed data, while the last three are for the observed data.

```

> flimMean(panss.flim,"Y", grouping="treat")
      1      2      3 1 obs 2 obs 3 obs
0 93.40000 91.40000 91.26000 93.40000 91.40000 91.26000
1 87.84636 93.54380 80.80000 87.87755 93.79592 80.80000
2 86.60223 92.80858 78.64123 84.47727 88.42105 77.20000
4 87.91717 95.21459 74.21225 83.92500 86.30000 71.63158
6 85.80121 104.10217 73.76264 75.67857 91.52174 69.09091
8 83.48531 103.63105 78.45876 73.72000 86.87500 71.66667
    
```

The mean responses can be plotted with the plot function – we specify the grouping argument so that separate curves are drawn for each treatment group:

```

> plot(panss.flim,"Y", grouping="treat", col=c("green","blue","brown"))
Mean response plot created for variable: Y
Full drawn: observed mean response.
Stipulated: flim hypothetical mean response.
    
```

Response variable separated by: treat which has 3 levels.
 Group: 1 has color green. Group: 2 has color blue. Group: 3 has color brown.

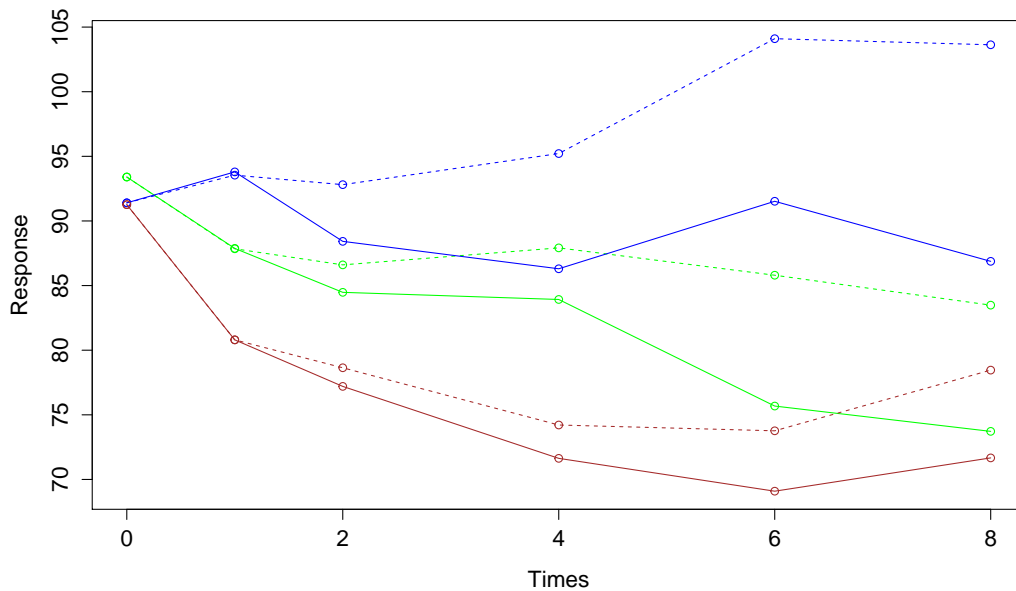


Figure 1: Observed (solid line) and hypothetical (dotted line) mean responses for haloperidol (green), placebo (blue) and risperidone (brown) treatment regimens among patients with schizophrenia.

The different treatments are 1: haloperidol, 2: placebo and 3: risperidone. Legends are not auto generated by the function and must be added by the user if needed. Observed and hypothetical mean responses are shown in Figure 1. Looking at the figure, we find that the haloperidol and risperidone treatment groups show better progress than the placebo group. Moreover, the risperidone group shows fewer symptoms than the haloperidol group. But the most striking feature is that among all

treatment groups, there are clear differences between the observed and the hypothetical curves. When considering the observed mean responses, one might infer that PANSS scores improve (decrease) over time regardless of treatment group. Indeed, even the placebo group shows a decreasing observed mean response. On the other hand, when the hypothetical mean responses are considered, the linear increments model suggests that this is due to informative drop-out, and therefore the observed data are underestimating the PANSS score. An increase in symptoms is even suggested in the placebo group, while the haloperidol and risperidone groups show a reduced effect (less decline) when comparing hypothetical to observed mean responses.

By default, the plot function displays the mean response – to show individual trajectories, use `ptype="spa"`. The spaghetti plot is shown in Figure 2, in which solid lines are used before drop out, and the dotted lines are used thereafter, when values are imputed:

```
plot(panss.flim,"Y", grouping="treat", col=c("green","blue","brown"), ptype="spa")
```

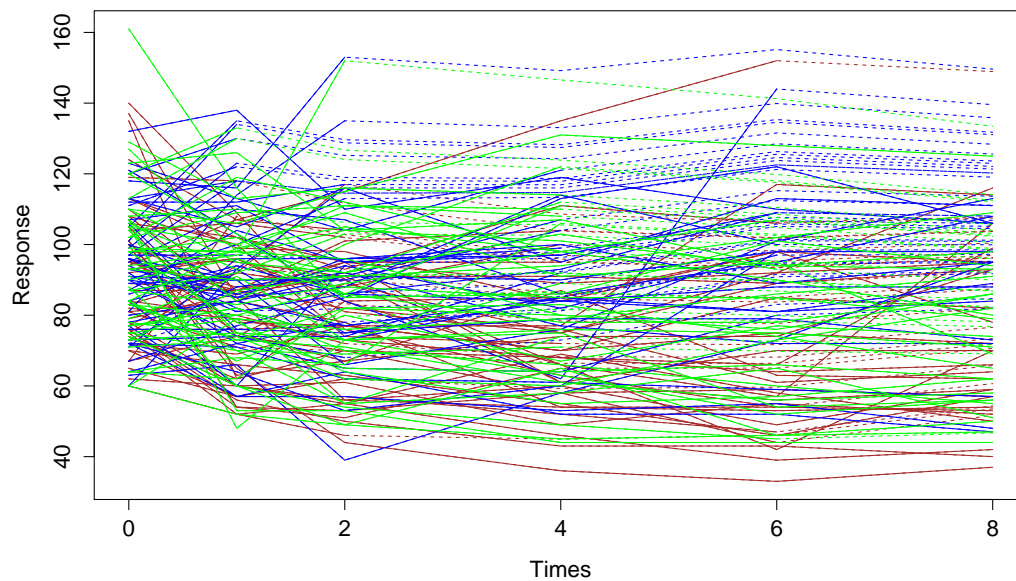


Figure 2: Individual responses for haloperidol, placebo and risperidone treatment regimens among patients with schizophrenia. Dotted lines are used after drop-out.

Assessing model fit

The fitted linear models for the increments are contained in the `flim` object as a list in the value `fit`. Users may perform standard model checking by directly accessing these. However, to view summary of the coefficients and running model diagnostics on the different time points in a more compact and convenient way, the function `flimList` is useful:

```
> flimList(panss.flim)
Call:
  flimList(panss.flim)
  Model: Y.inc ~ Y + factor(treat)
  Data: panss
  Times: 0 1 2 4 6

Coefficients:
(Intercept)      Y factor(treat)2 factor(treat)3
0      28.369 -0.363           6.971          -5.684
1       8.557 -0.112           1.145          -1.701
2      10.264 -0.103           1.732          -6.567
4       2.650 -0.054          11.399           0.923
6       6.199 -0.099           3.661           5.817
```

These coefficients represents the effects of responses and covariates on the expected change in the responses at the different time points.

The summary function will display the estimated coefficients, standard errors, t-values and p-values:

```
> summary(flimList(panss.flim))
Call:
  Model: Y.inc ~ Y + factor(treat)
  Data: panss
  Times: 0 1 2 4 6

Coefficients:

(Intercept)
  Estimate Std. Error  t value  Pr(>|t|)
0 28.369363  6.766822 4.1924203 4.796753e-05
1  8.557158  6.069304 1.4099075 1.610906e-01
2 10.263738  5.880129 1.7454954 8.385207e-02
4  2.650326  6.697694 0.3957073 6.933743e-01
6  6.199175  5.389000 1.1503387 2.542846e-01

Y
  Estimate Std. Error  t value  Pr(>|t|)
0 -0.36320130 0.06831664 -5.3164396 3.948642e-07
1 -0.11157312 0.06689537 -1.6678752 9.788447e-02
2 -0.10333217 0.06732753 -1.5347686 1.278774e-01
4 -0.05421329 0.07943341 -0.6824999 4.968945e-01
6 -0.09924193 0.06558725 -1.5131284 1.351699e-01

factor(treat)2
  Estimate Std. Error  t value  Pr(>|t|)
0  6.971035  3.174670 2.1958296 0.029704587
1  1.144598  3.002800 0.3811768 0.703729668
2  1.732388  3.049484 0.5680922 0.571196889
4 11.399145  4.024242 2.8326193 0.005839657
6  3.661010  3.599667 1.0170413 0.312963201

factor(treat)3
  Estimate Std. Error  t value  Pr(>|t|)
0 -5.6836117  3.160298 -1.7984418 0.07420171
1 -1.7008236  2.898236 -0.5868478 0.55838151
2 -6.5665452  2.891971 -2.2706126 0.02523154
4  0.9233581  3.712419  0.2487214 0.80421371
6  5.8172905  3.115758  1.8670547 0.06647537
```

In addition to `print` and `summary`, a `plot` function for `flimList` is available in the package. Plotting a `flimList` object will display the standard `lm` plot diagnostics in a 2x2 grid for each time point. These plots enable us to detect any major misspecification in the linear models for the increments. Diagnostics for the increment model at time point 0 for the PANSS data are shown in Figure 3.

The top left plot is showing the residuals vs fitted values, and should ideally be close to zero and not show any trend. The plot in the top right is a Q-Q plot that can reveal deviation from the normal assumption in the linear regression. The bottom left plot is typically used in addition to the first plot to check for misspecification and to check for heterogeneity in variances – users should note however that it might not be a problem with heterogeneity as we do not have to assume homogeneity of variances (Diggle et al., 2007). The lower right plot that shows residuals vs leverage is used to detect if some observations are affecting the model fit more strongly than others, users should look for any observation with large leverage and a big standardised residual (negative or positive), as this may distort the model.

To view the plots for the subsequent time points, users are prompted to press <ENTER>:

```
> plot(flimList(panss.flim), "Y")
  Model: Y.inc ~ Y + factor(treat)
Hit <Return> to see next plot:
Time: 0
```

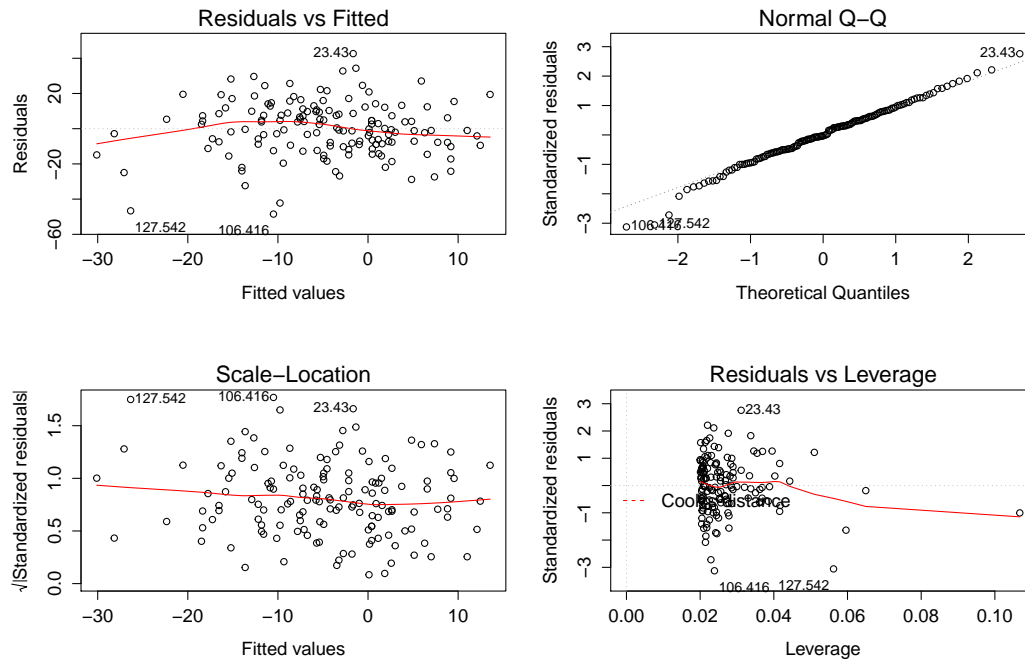



Figure 3: Plot diagnostics for linear fit at time point 0.

Bootstrapping

Bootstrapping is available once a "flim" object is fitted. Using the flimboot function, users have only to specify the number of bootstrap samples:

```
panss.boot <- flimboot(panss.flim, 100)
```

We can calculate the empirical standard errors for bootstrap estimates of the hypothetical mean response by:

```
> flimSD(panss.boot, "Y", grouping="treat")
```

To plot the hypothetical mean response from the original dataset together with point wise 95% bootstrap confidence bands, we can use the plot function for "flimboot" objects (Figure 4):

```
plot(panss.boot, "Y", "treat")
```

Artificial censoring and causal inference

As argued before, the linear increments model can be a potential method for doing causal inference. This is done by imputing counterfactual response values as if an intervention never took place; the imputed values can then be compared to the observed values to assess what effect the intervention had on the response.

We will demonstrate this feature with a simulated dataset that is made to mimic a study of patients with HIV. The patients in the study have steadily declining CD4 values, and as CD4 values continue to decrease, patients initiate treatment with increasing probability. Receiving treatment increases the patients' CD4 values. CD4 is also affecting the hazard of event as those with a low value are at greater risk of developing AIDS during the study. The event AIDS is simulated on each time point with an additive hazard model where CD4 status and treatment are affecting the hazard. If patients get AIDS, they are censored from the study.

Since CD4 is affecting both probability of starting treatment and the outcome, and treatment on the other hand is affecting CD4, it would be useful to know what the CD4 values would have been if patients counterfactually did not start treatment. Note that it is assumed that treatment is affecting the response value on the next observation after treatment starts, and then on consecutive observations. Once a patient starts treatment, he or she may not go back to being untreated.

It is simple to create counterfactual values as if treatment never started:

Hypothetical mean with bootstrap confidence bands

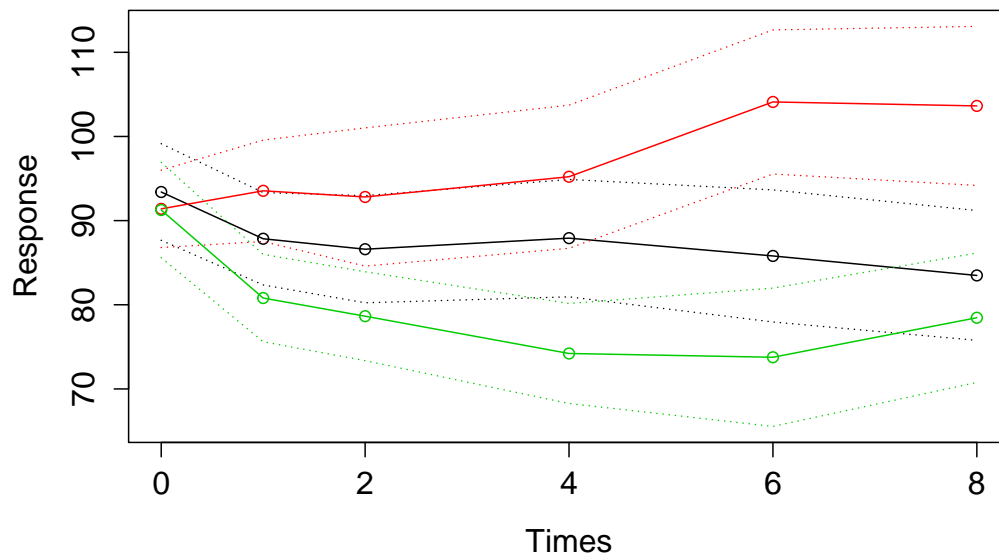


Figure 4: Hypothetical mean responses with bootstrap confidence bands.

```
data(CD4sim)
CD4.flim <- flim(cd4 ~ cd4, id="id", obstime="time", data=CD4sim,
               art.cens="treat")
```

The fitted object contains a dataset with the counterfactual CD4 values added in a separate column:

```
> head(CD4.flim$dataset)
  id time treat      cd4 AIDS      cd4.cf
1  1   0     0  7.071068    0  7.071068
2  1   1     0  6.084063    0  6.084063
3  1   2     0  5.471980    0  5.471980
4  1   3     0  4.644535    0  4.644535
5  1   4     0  3.900693    0  3.900693
6  1   5     0  3.262019    0  3.262019
```

This option is intended for advanced users and does not come with readily available tools for investigating results. In Figure 5 we have for illustration chosen to plot the mean response curves for observed and counterfactual CD4 values. Here we can see that the linear increments model suggests the CD4 values would have been lower if the treated patients were untreated.

Summary

The linear increments model is a simple approach for dealing with missing data in longitudinal studies, and FLIM provides a straightforward implementation with standard formula syntax. Using the package, we can recreate hypothetical response trajectories that could have been observed in the absence of missingness. The mean structure of the reconstructed data will be correct under certain assumptions (3); however, these imputed values will not exhibit the correct variability, and confidence bands for the mean response can be based on bootstrapping.

With a readily available tool for handling and imputing missing data, users should emphasise which assumptions need to be made in order for the hypothetical developments to be justifiable. Indeed, in some sense imputation is a manipulation of the data, and this is certainly true when one is considering averages such as the mean response. Uncritically applying the linear increments model to obtain a "complete" dataset is not advised.

Knowing when the required assumptions are met can be a challenge, and the issue is further complicated when missingness is nonmonotone. The basic message is that the method is applicable

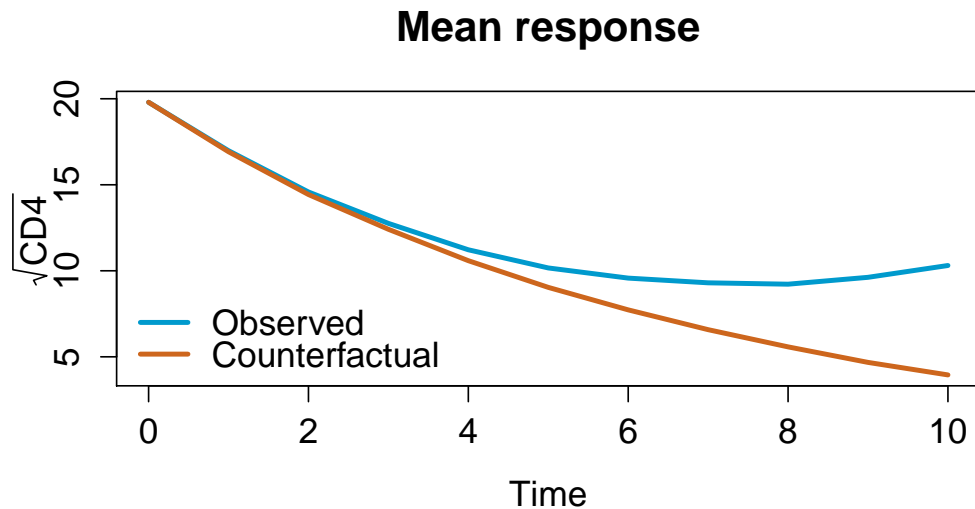


Figure 5: Mean response for observed and counterfactual CD4 values

when the observed increments are representative of those from the general population; this goes both for monotone and nonmonotone missing data, but may be more difficult to establish in the latter case.

Bibliography

- O. O. Aalen and N. Gunnes. A dynamic approach for reconstructing missing longitudinal data using the linear increments model. *Biostatistics*, 11:453–472, 2010. [p137, 138, 139]
- O. O. Aalen, K. Røysland, J. M. Gran, and B. Ledergerber. Causality, mediation and time: a dynamic viewpoint. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 175(4):831–861, 2012. [p141]
- P. Diggle. Dealing with missing values in longitudinal studies. In *Recent Advances in the Statistical Analysis of Medical Data*, editor, B. S. Everitt and G. Dunn, pages 203–228, 1998. [p142, 143]
- P. Diggle, D. M. Farewell, and R. Henderson. Analysis of longitudinal data with drop-out: objectives, assumptions and a proposal. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 56: 499–550, 2007. [p137, 138, 139, 146]
- D. M. Farewell. *Linear models for censored data*. PhD thesis, Lancaster University, 2006. [p137]
- N. Gunnes, D. M. Farewell, O. O. Aalen, and T. G. Seierstad. Analysis of censored discrete longitudinal data: estimation of mean response. *Statistics in Medicine*, 28:605–624, 2009. [p139]
- M. A. Hernán, E. Lanoy, D. Costagliola, and J. M. Robins. Comparison of dynamic treatment regimes via inverse probability weighting. *Basic & Clinical Pharmacology & Toxicology*, 98:237–242 2006. [p140]
- J. W. Hogan, J. Roy, and C. Korkontzelou. Handling drop-out in longitudinal studies. *Statistics in Medicine*, 23:1455–1497, 2004. [p137]
- A. Zeileis and G. Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL <http://www.jstatsoft.org/v14/i06/>. [p142]

Rune Hoff
 Oslo Centre for Biostatistics and Epidemiology
 Department of Biostatistics
 University of Oslo
 Norway
rune.hoff@medisin.uio.no

Jon Michael Gran
Oslo Centre for Biostatistics and Epidemiology
Department of Biostatistics
University of Oslo
Norway
j.m.gran@medisin.uio.no

Daniel Farewell
Cochrane Institute of Primary Care & Public Health
School of Medicine
Cardiff University
Wales
FarewellD@cf.ac.uk

MVN: An R Package for Assessing Multivariate Normality

by Selcuk Korkmaz, Dincer Goksuluk and Gokmen Zararsiz

Abstract Assessing the assumption of multivariate normality is required by many parametric multivariate statistical methods, such as MANOVA, linear discriminant analysis, principal component analysis, canonical correlation, etc. It is important to assess multivariate normality in order to proceed with such statistical methods. There are many analytical methods proposed for checking multivariate normality. However, deciding which method to use is a challenging process, since each method may give different results under certain conditions. Hence, we may say that there is no best method, which is valid under any condition, for normality checking. In addition to numerical results, it is very useful to use graphical methods to decide on multivariate normality. Combining the numerical results from several methods with graphical approaches can be useful and provide more reliable decisions. Here, we present an R package, **MVN**, to assess multivariate normality. It contains the three most widely used multivariate normality tests, including Mardia's, Henze-Zirkler's and Royston's, and graphical approaches, including chi-square Q-Q, perspective and contour plots. It also includes two multivariate outlier detection methods, which are based on robust Mahalanobis distances. Moreover, this package offers functions to check the univariate normality of marginal distributions through both tests and plots. Furthermore, especially for non-R users, we provide a user-friendly web application of the package. This application is available at <http://www.biosoft.hacettepe.edu.tr/MVN/>.

Introduction

Many multivariate statistical analysis methods, such as MANOVA and linear discriminant analysis (**MASS**, Venables and Ripley (2002)), principal component analysis (**FactoMineR**, Husson et al. (2014), **psych**, Revelle (2014)), and canonical correlation (**CCA**, González and Déjean (2012)), require multivariate normality (MVN) assumption. If the data are multivariate normal (exactly or approximately), such multivariate methods provide more reliable results. The performances of these methods dramatically decrease if the data are not multivariate normal. Hence, researchers should check whether data are multivariate normal or not before continuing with such parametric multivariate analyses.

Many statistical tests and graphical approaches are available to check the multivariate normality assumption. Burdinski (2000) reviewed several statistical and practical approaches, including the Q-Q plot, box-plot, stem and leaf plot, Shapiro-Wilk, and Kolmogorov-Smirnov tests to evaluate univariate normality, contour and perspective plots for assessing bivariate normality, and the chi-square Q-Q plot to check multivariate normality. The author demonstrated each procedure using real data from George and Mallery (1999). Ramzan et al. (2013) reviewed numerous graphical methods for assessing both univariate and multivariate normality and showed their use in a real life problem to check the MVN assumption using chi-square and beta Q-Q plots. Holgersson (2006) stated the importance of graphical procedures and presented a simple graphical tool, which is based on the scatter plot of two correlated variables to assess whether the data belong to a multivariate normal distribution or not. Svantesson and Wallace (2003) applied Royston's and Henze-Zirkler's tests to multiple-input multiple-output data to test MVN. According to the review by Mecklin and Mundfrom (2005), more than fifty statistical methods are available for testing MVN. They conducted a comprehensive simulation study based on type I and type II error and concluded that no single test excelled in all situations. The authors suggested using Henze-Zirkler's and Royston's tests among others for assessing MVN because of their good type I error control and power. Moreover, to diagnose the reason for deviation from multivariate normality, the authors suggested the use of Mardia's multivariate skewness and kurtosis statistics test as well as graphical approaches such as the chi-square Q-Q plot. Deciding which test to use can be a daunting task for researchers (mostly for non-statisticians) and it is very useful to perform several tests and examine the graphical methods simultaneously. Although there are a number of studies describing multifarious approaches, there is no single easy-to-use, up-to-date and comprehensive tool to apply various statistical tests and graphical methods together at present.

In this paper, we introduce an R package, **MVN** (Korkmaz et al., 2014), which implements the three most widely used MVN tests, including Mardia's, Henze-Zirkler's, and Royston's. In addition to statistical tests, the **MVN** also provides some graphical approaches such as chi-square Q-Q, perspective, and contour plots. Moreover, this package includes two multivariate outlier detection methods, which are based on Mahalanobis distance. In addition to multivariate normality, users can also check univariate normality tests and plots to diagnose deviation from normality via package version 3.7 and later. First, we discuss the theoretical background on the corresponding MVN tests. Second, two illustrative examples are presented in order to demonstrate the applicability of the package. Finally,

we present a newly developed web interface of the **MVN** package, which can be especially handy for non-R users. The R version of **MVN** is publicly available from the Comprehensive R Archive Network (CRAN, <http://CRAN.R-project.org/package=MVN>).

Multivariate normality tests

Mardia’s MVN test

Mardia (1970) proposed a multivariate normality test which is based on multivariate extensions of skewness ($\hat{\gamma}_{1,p}$) and kurtosis ($\hat{\gamma}_{2,p}$) measures as follows:

$$\hat{\gamma}_{1,p} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n m_{ij}^3 \quad \text{and} \quad \hat{\gamma}_{2,p} = \frac{1}{n} \sum_{i=1}^n m_{ii}^2 \tag{1}$$

where $m_{ij} = (x_i - \bar{x})' S^{-1} (x_j - \bar{x})$, the squared Mahalanobis distance, and p is the number of variables. The test statistic for skewness, $(n/6)\hat{\gamma}_{1,p}$, is approximately χ^2 distributed with $p(p+1)(p+2)/6$ degrees of freedom. Similarly, the test statistic for kurtosis, $\hat{\gamma}_{2,p}$, is approximately normally distributed with mean $p(p+2)$ and variance $8p(p+2)/n$.

For small samples, the power and the type I error could be violated. Therefore, Mardia (1974) introduced a correction term into the skewness test statistic, usually when $n < 20$, in order to control type I error. The corrected skewness statistic for small samples is $(nk/6)\hat{\gamma}_{1,p}$, where $k = (p+1)(n+1)(n+3)/(n(n+1)(p+1) - 6)$. This statistic is also distributed as χ^2 with degrees of freedom $p(p+1)(p+2)/6$.

Henze-Zirkler’s MVN test

The Henze-Zirkler’s test is based on a non-negative functional distance that measures the distance between two distribution functions. If data are distributed as multivariate normal, the test statistic is approximately log-normally distributed. First, the mean, variance, and smoothness parameter are calculated. Then, the mean and the variance are log-normalized and the p-value is estimated (Henze and Zirkler, 1990; Johnson and Wichern, 1992; Henze and Wagner, 1997; Mecklin and Mundfrom, 2003; Alpar, 2013). The test statistic of Henze-Zirkler’s multivariate normality test is given in equation 2.

$$HZ = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n e^{-\frac{\beta^2}{2} D_{ij}} - 2 \left(1 + \beta^2\right)^{-\frac{p}{2}} \sum_{i=1}^n e^{-\frac{\beta^2}{2(1+\beta^2)} D_i} + n \left(1 + 2\beta^2\right)^{-\frac{p}{2}} \tag{2}$$

where

$$\begin{aligned} p &: \text{ number of variables} \\ \beta &= \frac{1}{\sqrt{2}} \left(\frac{n(2p+1)}{4} \right)^{\frac{1}{p+4}} \\ D_{ij} &= (x_i - x_j)' S^{-1} (x_i - x_j) \\ D_i &= (x_i - \bar{x})' S^{-1} (x_i - \bar{x}) = m_{ii} \end{aligned}$$

From equation 2, D_i gives the squared Mahalanobis distance of i^{th} observation to the centroid and D_{ij} gives the Mahalanobis distance between i^{th} and j^{th} observations. If data are multivariate normal, the test statistic (HZ) is approximately log-normally distributed with mean μ and variance σ^2 as given below:

$$\begin{aligned} \mu &= 1 - \frac{a^{-\frac{p}{2}} \left(1 + p\beta^{\frac{2}{a}} + (p(p+2)\beta^4)\right)}{2a^2} \\ \sigma^2 &= 2 \left(1 + 4\beta^2\right)^{-\frac{p}{2}} + \frac{2a^{-p} (1 + 2p\beta^4)}{a^2} + \frac{3p(p+2)\beta^8}{4a^4} \\ &\quad - 4w_\beta^{-\frac{p}{2}} \left(1 + \frac{3p\beta^4}{2w_\beta} + \frac{p(p+2)\beta^8}{2w_\beta^2}\right) \end{aligned}$$

where $a = 1 + 2\beta^2$ and $w_\beta = (1 + \beta^2)(1 + 3\beta^2)$. Hence, the log-normalized mean and variance of the

HZ statistic can be defined as follows:

$$\log(\mu) = \log\left(\sqrt{\frac{\mu^4}{\sigma^2 + \mu^2}}\right) \quad \text{and} \quad \log(\sigma^2) = \log\left(\frac{\sigma^2 + \mu^2}{\sigma^2}\right) \quad (3)$$

By using the log-normal distribution parameters, μ and σ , we can test the significance of multivariate normality. The *Wald* test statistic for multivariate normality is given in equation 4.

$$z = \frac{\log(HZ) - \log(\mu)}{\log(\sigma)} \quad (4)$$

Royston's MVN test

Royston's test uses the *Shapiro-Wilk/Shapiro-Francia* statistic to test multivariate normality. If kurtosis of the data is greater than 3, then it uses the *Shapiro-Francia* test for leptokurtic distributions, otherwise it uses the *Shapiro-Wilk* test for platykurtic distributions (Shapiro and Wilk, 1964; Royston, 1982, 1983, 1992; Johnson and Wichern, 1992; Royston, 1995; Mecklin and Mundfrom, 2005).

Let W_j be the *Shapiro-Wilk/Shapiro-Francia* test statistic for the j^{th} variable ($j = 1, 2, \dots, p$) and Z_j be the values obtained from the normality transformation proposed by Royston (1992).

$$\begin{aligned} \text{if } 4 \leq n \leq 11; & \quad x = n \quad \text{and} \quad w_j = -\log\left[\gamma - \log(1 - W_j)\right] \\ \text{if } 12 \leq n \leq 2000; & \quad x = \log(n) \quad \text{and} \quad w_j = \log(1 - W_j) \end{aligned} \quad (5)$$

As seen from equation 5, x and w_j -s change with the sample size (n). By using equation 5, transformed values of each random variable can be obtained from equation 6.

$$Z_j = \frac{w_j - \mu}{\sigma} \quad (6)$$

where γ , μ and σ are derived from the polynomial approximations given in equation 7. The polynomial coefficients are provided by Royston (1992) for different sample sizes.

$$\begin{aligned} \gamma &= a_{0\gamma} + a_{1\gamma}x + a_{2\gamma}x^2 + \dots + a_{d\gamma}x^d \\ \mu &= a_{0\mu} + a_{1\mu}x + a_{2\mu}x^2 + \dots + a_{d\mu}x^d \\ \log(\sigma) &= a_{0\sigma} + a_{1\sigma}x + a_{2\sigma}x^2 + \dots + a_{d\sigma}x^d \end{aligned} \quad (7)$$

The Royston's test statistic for multivariate normality is as follows:

$$H = \frac{e^{\sum_{j=1}^p \psi_j}}{p} \sim \chi_e^2 \quad (8)$$

where e is the equivalent degrees of freedom (edf) and $\Phi(\cdot)$ is the cumulative distribution function for the standard normal distribution such that,

$$\begin{aligned} e &= p/[1 + (p - 1)\bar{c}] \\ \psi_j &= \left\{ \Phi^{-1} \left[\Phi(-Z_j)/2 \right] \right\}^2, \quad j = 1, 2, \dots, p. \end{aligned} \quad (9)$$

As seen from equation 9, another extra term \bar{c} has to be calculated in order to continue with the statistical significance of Royston's test statistic given in equation 8. Let R be the correlation matrix and r_{ij} be the correlation between i^{th} and j^{th} variables. Then, the extra term \bar{c} can be found by using equation 10.

$$\bar{c} = \sum_i \sum_j \frac{c_{ij}}{p(p-1)}, \quad \{c_{ij}\}_{i \neq j} \quad (10)$$

where

$$c_{ij} = \begin{cases} g(r_{ij}, n) & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

with the boundaries of $g(\cdot)$ as $g(0, n) = 0$ and $g(1, n) = 1$. The function $g(\cdot)$ is defined as follows:

$$g(r, n) = r^\lambda \left[1 - \frac{\mu}{\nu} (1 - r)^\mu \right]$$

The unknown parameters μ , λ , and ν were estimated from a simulation study conducted by Ross (1980). He found $\mu = 0.715$ and $\lambda = 5$ for sample size $10 \leq n \leq 2000$ and ν is a cubic function which

can be obtained as follows:

$$v(n) = 0.21364 + 0.015124x^2 - 0.0018034x^3$$

where $x = \log(n)$.

Implementation of MVN package

The **MVN** package contains several functions in the S4 class system. The data to be analyzed should be given as a "data.frame" or "matrix" object. In this example, we will work with the famous *Iris* data set. These data are from a multivariate data set introduced by Fisher (1936) as an application of linear discriminant analysis. It is also called Anderson's *Iris* data set because Edgar Anderson collected the data to measure the morphologic variation of *Iris* flowers of three related species (Anderson, 1936). First of all, the **MVN** package should be loaded in order to use related functions.

```
# load MVN package
library(MVN)
```

Similarly, the *Iris* data can be loaded from the R database by using the following R code:

```
# load Iris data
data(iris)
```

The *Iris* data set consists of 150 samples from each of the three species of *Iris* including *setosa*, *virginica*, and *versicolor*. For each sample, four variables were measured including the length and width of the *sepals* and *petals*, in centimeters.

Example I: For simplicity, we will work with a subset of these data which contain only 50 samples of *setosa* flowers, and check MVN assumption using Mardia's, Royston's and Henze-Zirkler's tests.

```
# setosa subset of the Iris data
setosa <- iris[1:50, 1:4]
```

Mardia's MVN test: mardiaTest(...)

The `mardiaTest` function is used to calculate the Mardia's test multivariate skewness and kurtosis coefficients as well as their corresponding statistical significance. This function can also calculate the corrected version of the skewness coefficient for small sample size ($n < 20$).

```
result <- mardiaTest(setosa, qqplot = FALSE)
result
```

```
-----
Mardia's Multivariate Normality Test
-----
data : setosa

g1p          : 3.079721
chi.skew     : 25.66434
p.value.skew : 0.1771859

g2p          : 26.53766
z.kurtosis   : 1.294992
p.value.kurt : 0.1953229

chi.small.skew : 27.85973
p.value.small  : 0.1127617

Result       : Data are multivariate normal.
-----
```

Here:

g1p: Mardia's estimate of multivariate skewness, i.e. $\hat{\gamma}_{1,p}$ given in equation 1,
 chi.skew: test statistic for multivariate skewness,
 p.value.skew: significance value of skewness statistic,

`g2p`: Mardia's estimate of multivariate kurtosis, i.e. $\hat{\gamma}_{2,p}$ given in equation 1,
`z.kurtosis`: test statistic for multivariate kurtosis,
`p.value.kurt`: significance value of kurtosis statistic,
`chi.small.skew`: test statistic for multivariate skewness with small sample correction,
`p.value.small`: significance value of small sample skewness statistic.

As seen from the results given above, both the skewness ($\hat{\gamma}_{1,p} = 3.0797, p = 0.1772$) and kurtosis ($\hat{\gamma}_{2,p} = 26.5377, p = 0.1953$) estimates indicate multivariate normality. Therefore, according to Mardia's MVN test, this data set follows a multivariate normal distribution.

Henze-Zirkler's MVN test: `hzTest(...)`

One may use the `hzTest` function in the `MVN` package to perform the Henze-Zirkler's test.

```

result <- hzTest(setosa, qqplot = FALSE)
result

  Henze-Zirkler's Multivariate Normality Test
-----
data : setosa

HZ      : 0.9488453
p-value : 0.04995356

Result  : Data are not multivariate normal.
-----
  
```

Here, HZ is the value of the Henze-Zirkler's test statistic at significance level 0.05 and p-value is the significance value of this test statistic, i.e., the significance of multivariate normality. As the p-value, which is derived from `hzTest`, is mathematically lower than 0.05, one can conclude that this multivariate data set deviates slightly from multivariate normality ($HZ = 0.9488, p = 0.04995$). As the p-value is very close to 0.05, researchers should also check the multivariate graphical approaches as well as univariate tests and plots to make a more reliable decision on multivariate normality.

Royston's MVN test: `roystonTest(...)`

In order to carry out the Royston's test, `roystonTest` function in the `MVN` package can be used as follows:

```

result <- roystonTest(setosa, qqplot = FALSE)
result

  Royston's Multivariate Normality Test
-----
data : setosa

H      : 31.51803
p-value : 2.187653e-06

Result  : Data are not multivariate normal.
-----
  
```

Here, H is the value of the Royston's test statistic at significance level 0.05 and p-value is an approximate significance value for the test with respect to edf. According to Royston's test, the *setosa* data set does not appear to follow a multivariate normal distribution ($H = 31.518, p < 0.001$).

Chi-square Q-Q plot

One can clearly see that different MVN tests may come up with different results. MVN assumption was rejected by Henze-Zirkler's and Royston's tests; however, it was not rejected by Mardia's test at a significance level of 0.05. In such cases, examining MVN plots along with hypothesis tests can be quite useful in order to reach a more reliable decision.

The Q-Q plot, where “Q” stands for quantile, is a widely used graphical approach to evaluate the agreement between two probability distributions. Each axis refers to the quantiles of probability distributions to be compared, where one of the axes indicates theoretical quantiles (hypothesized quantiles) and the other indicates the observed quantiles. If the observed data fit hypothesized distribution, the points in the Q-Q plot will approximately lie on the line $y = x$.

MVN has the ability to create three multivariate plots. One may use the `qqplot = TRUE` option in the `mardiaTest`, `hzTest`, and `roystonTest` functions to create a chi-square Q-Q plot. We can create this plot for the *setosa* data set to see whether there are any deviations from multivariate normality. Figure 1 shows the chi-square Q-Q plot of the first 50 rows of the *Iris* data, which are *setosa* flowers. It can be seen from Figure 1 that there are some deviations from the straight line and this indicates possible departures from a multivariate normal distribution.

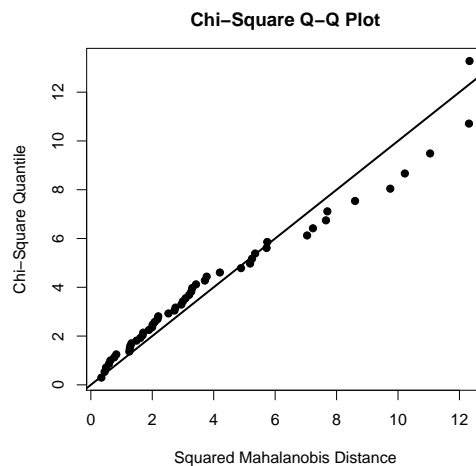


Figure 1: Chi-Square Q-Q plot for *setosa* data set.

As a result, we can conclude that this data set does not satisfy MVN assumption based on the fact that the two test results are against it and the chi-square Q-Q plot indicates departures from multivariate normal distribution.

Univariate plots and tests

As noted by several authors (Burdenski, 2000; Stevens, 2012; Kass et al., 2014), if data have a multivariate normal distribution, then, each of the variables has a univariate normal distribution; but the opposite does not have to be true. Hence, checking univariate plots and tests could be very useful to diagnose the reason for deviation from MVN. We can check this assumption through `uniPlot` and `uniNorm` functions from the package. The `uniPlot` function is used to create univariate plots, such as Q-Q plots (Figure 2a), histograms with normal curves (Figure 2b), box-plots and scatterplot matrices.

```
uniPlot(setosa, type = "qqplot") # draw univariate Q-Q plots
uniPlot(setosa, type = "histogram") # draw univariate histograms
```

As seen from Figure 2, *Petal.Width* has a right-skewed distribution whereas other variables have approximately normal distributions. Thus, we can conclude that problems with multivariate normality arise from the skewed distribution of *Petal.Width*. In addition to the univariate plots, one can also perform univariate normality tests using the `uniNorm` function. It provides several widely used univariate normality tests, including Shapiro-Wilk, Cramer-von Mises, Lilliefors, and Anderson-Darling. For example, the following code chunk is used to perform the Shapiro-Wilk’s normality test on each variable:

```
uniNorm(setosa, type = "SW")
```

Shapiro-Wilk's test of Normality

Variable	Statistic	p-value	Normality
1 Sepal.Length	0.9777	0.4595	YES
2 Sepal.Width	0.9717	0.2715	YES
3 Petal.Length	0.9550	0.0548	YES
4 Petal.Width	0.7998	0.0000	NO

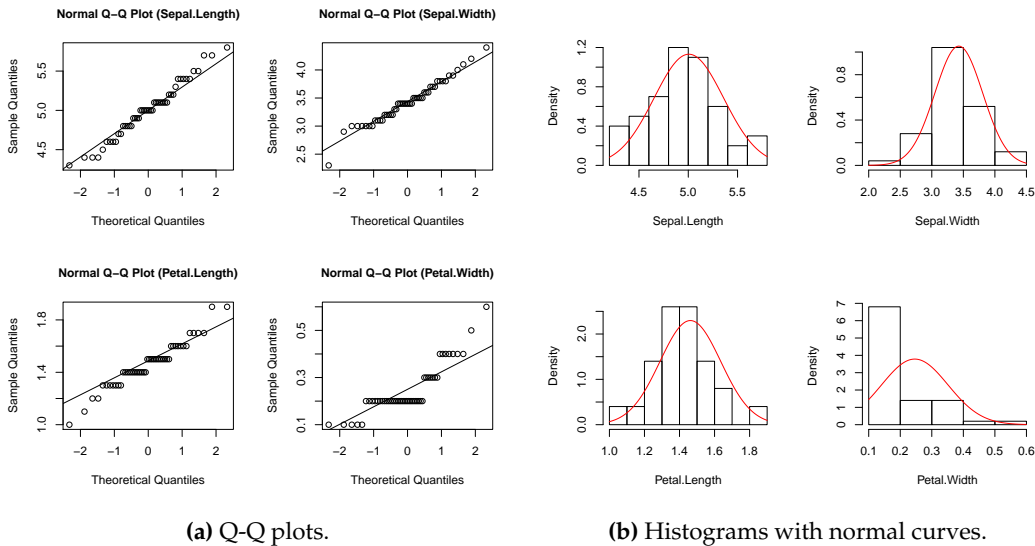


Figure 2: Univariate plots of *setosa*.

From the above results, we can see that all variables, except *Petal.Width* in the *setosa* data set, have univariate normal distributions at significance level 0.05. We can now drop *Petal.Width* from *setosa* data and recheck the multivariate normality. MVN results are given in Table 1.

Test	Test Statistic	p-value
Mardia		
Skewness	11.249	0.338
Kurtosis	1.287	0.198
Henze-Zirkler	0.524	0.831
Royston	7.255	0.060

Table 1: MVN test results (*setosa* without *Petal.Width*).

According to the three MVN test results in Table 1, *setosa* without *Petal.Width* has a multivariate normal distribution at significance level 0.05.

Example II: Whilst the Q-Q plot is a general approach for assessing MVN in all types of numerical multivariate datasets, perspective and contour plots can only be used for bivariate data. To demonstrate the applicability of these two approaches, we will use a subset of *Iris* data, named *setosa2*, including the *sepal length* and *sepal width* variables of the *setosa* species.

Perspective and contour plots

Univariate normal marginal densities are a necessary but not a sufficient condition for MVN. Hence, in addition to univariate plots, creating perspective and contour plots will be useful. The perspective plot is an extension of the univariate probability distribution curve into a 3-dimensional probability distribution surface related with bivariate distributions. It also gives information about where data are gathered and how two variables are correlated with each other. It consists of three dimensions where two dimensions refer to the values of the two variables and the third dimension, which like in univariate cases, is the value of the multivariate normal probability density function. Another alternative graph, which is called the “contour plot”, involves the projection of the perspective plot into a 2-dimensional space and this can be used for checking multivariate normality assumption. For bivariate normally distributed data, we expect to obtain a three-dimensional bell-shaped graph from the perspective plot. Similarly, in the contour plot, we can observe a similar pattern.

To construct a perspective and contour plot for Example 2, we can use the `mvnPlot` function in `MVN`. This function requires an object of the “`MVN`” class that is the result from one of the `MVN` functions. In the following codes, the object from `hzTest` is used for the perspective plot given in Figure 3a. It is also possible to create a contour plot of the data. Contour graphs are very useful as they give information about normality and correlation at the same time. Figure 3b shows the contour plot of *setosa* flowers. As can be seen from the graph, this is simply a top view of the perspective plot where

the third dimension is represented with ellipsoid contour lines. From this graph, we can say that there is a positive correlation among the *sepal* measures of flowers since the contour lines lie around the main diagonal. If the correlation were zero, the contour lines would be circular rather than ellipsoid.

```
setosa2 <- iris[1:50, 1:2]
result <- hzTest(setosa2, qqplot=FALSE)
mvnPlot(result, type = "persp", default = TRUE) # draw a perspective plot
mvnPlot(result, type = "contour", default = TRUE) # draw a contour plot
```

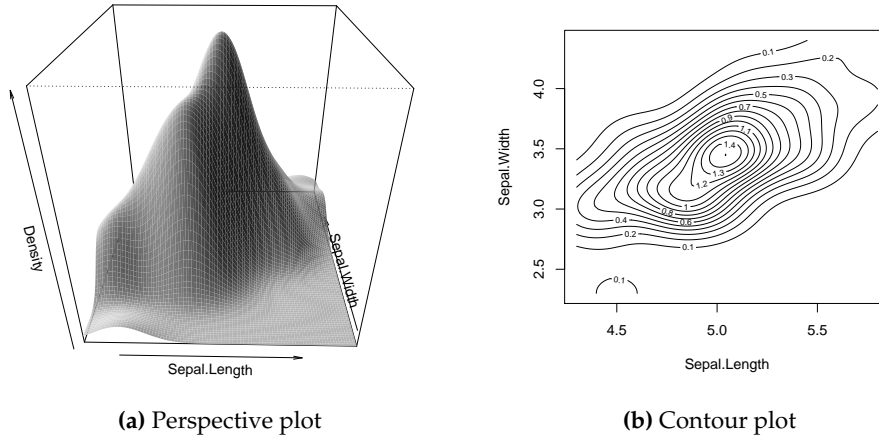


Figure 3: Perspective and contour plot for bivariate *setosa2* data set.

As neither the univariate plots in Figure 2 nor the multivariate plots in Figure 3 show any significant deviation from MVN, we can now perform the MVN tests to evaluate the statistical significance of bivariate normal distribution of the *setosa2* data set.

Test	Test Statistic	p-value
Mardia		
Skewness	0.760	0.944
Kurtosis	0.093	0.926
Henze-Zirkler	0.286	0.915
Royston	2.698	0.245

Table 2: MVN test results (*setosa* with *sepal* measures).

All three tests in Table 2 indicate that the data set satisfies bivariate normality assumption at the significance level 0.05. Moreover, the perspective and contour plots are in agreement with the test results and indicate approximate bivariate normality.

Figures 3a and 3b were drawn using a pre-defined graphical option by the authors. However, users may change these options by setting function entry to `default = FALSE`. If the default is FALSE, optional arguments from the `plot`, `persp` and `contour` functions may be introduced to the corresponding graphs.

Multivariate outliers

Multivariate outliers are the common reason for violating MVN assumption. In other words, MVN assumption requires the absence of multivariate outliers. Thus, it is crucial to check whether the data have multivariate outliers, before starting multivariate analysis. The MVN includes two multivariate outlier detection methods which are based on robust Mahalanobis distances ($rMD(x)$). Mahalanobis distance is a metric which calculates how far each observation is to the center of joint distribution, which can be thought of as the centroid in multivariate space. Robust distances are estimated from minimum covariance determinant estimators rather than the sample covariance (Rousseeuw and Leroy, 1987). These two approaches, defined as Mahalanobis distance and adjusted Mahalanobis distance in the package, detect multivariate outliers as given below,

Mahalanobis Distance:

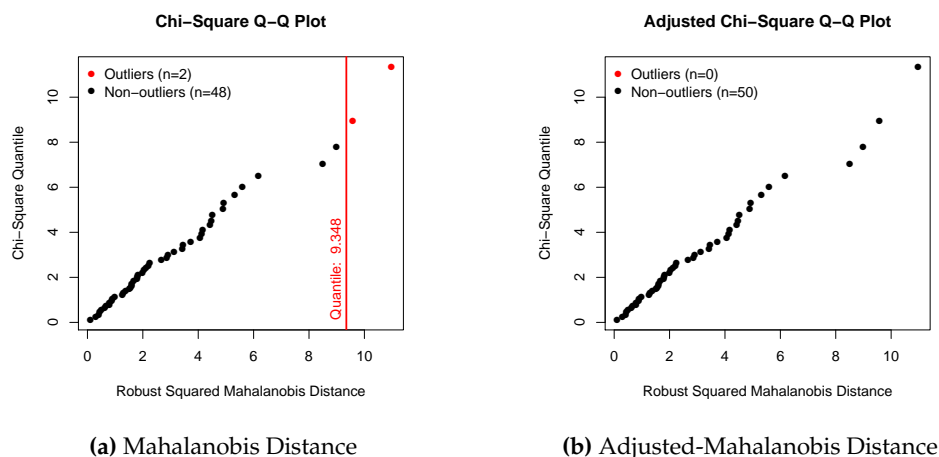
1. Compute robust Mahalanobis distances ($rMD(x_i)$),
2. Compute the 97.5 percent quantile (Q) of the chi-square distribution,
3. Declare $rMD(x_i) > Q$ as possible outlier.

Adjusted Mahalanobis Distance:

1. Compute robust Mahalanobis distances ($rMD(x_i)$),
2. Compute the 97.5 percent adjusted quantile (AQ) of the chi-Square distribution,
3. Declare $rMD(x_i) > AQ$ as possible outlier.

The `mvOutlier` function is used to detect multivariate outliers as given below. It also returns a new data set in which declared outliers are removed. Moreover, Q-Q plots can be created by setting `qqplot = TRUE` within `mvOutlier` for visual inspection of the possible outliers. For this example, we will use another subset of the *Iris* data, which is *versicolor* flowers, with the first three variables.

```
versicolor <- iris[51:100, 1:3]
# Mahalanobis distance
result <- mvOutlier(versicolor, qqplot = TRUE, method = "quan")
# Adjusted Mahalanobis distance
result <- mvOutlier(versicolor, qqplot = TRUE, method = "adj.quan")
```



(a) Mahalanobis Distance

(b) Adjusted-Mahalanobis Distance

Figure 4: Multivariate outlier detection.

From Figure 4, Mahalanobis distance declares 2 observations as multivariate outlier whereas adjusted Mahalanobis distance declares none. See Filzmoser et al. (2005) for further information on multivariate outliers.

Web interface for the MVN package

The purpose of the package is to provide MVN tests along with graphical approaches for assessing MVN. Moreover, this package offers univariate tests and plots, and multivariate outlier detection for checking MVN assumptions through R. However, using R codes might be challenging for new R users. Therefore, we also developed a user-friendly web application of MVN by using [shiny](http://www.rstudio.com/shiny/)¹ (RStudio, Inc., 2014), and this is publicly available.² This web-tool, which is an interactive application, has all the features that the MVN package has.

To start analysis, users need to upload or paste their data to the web-tool as described in the *Data upload* tab. Three example data sets are available on this page for researchers to test the tool and prepare their data in an appropriate form (Figure 5a). Before performing MVN tests, the univariate normality assumption can be checked through univariate plots (Q-Q plot, histograms with normal curve, box-plot, and scatterplot matrix) and tests (Shapiro-Wilk, Cramer-von Mises, Lilliefors, and

¹<http://www.rstudio.com/shiny/>

²<http://www.biosoft.hacettepe.edu.tr/MVN>

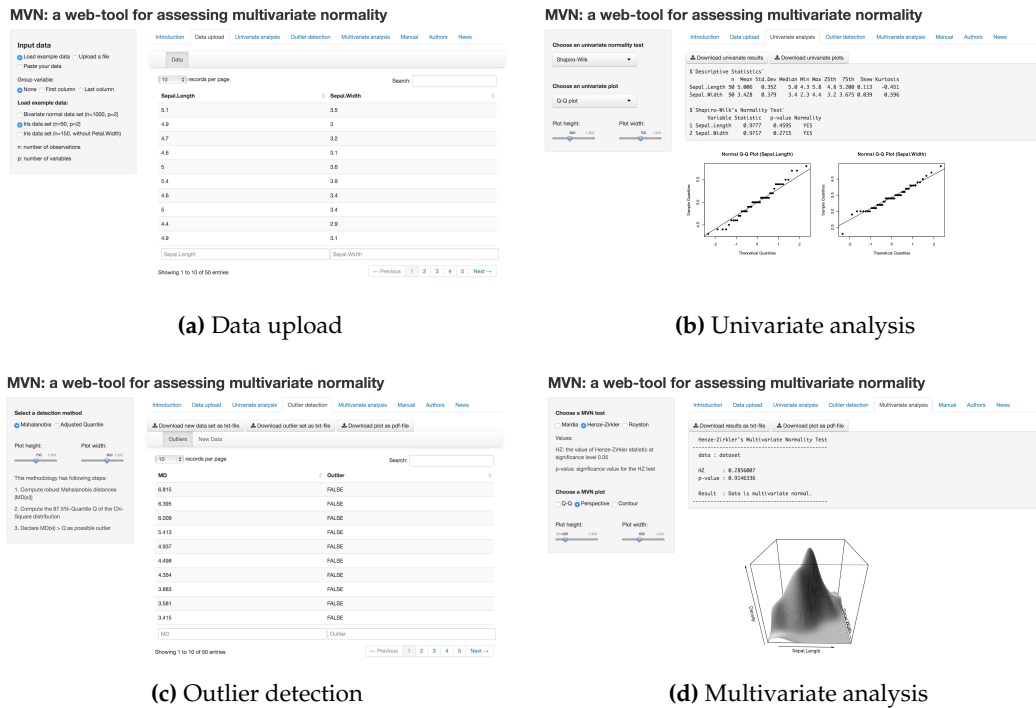


Figure 5: Web interface of the MVN package.

Anderson-Darling) in the *Univariate analysis* tab (Figure 5b) and possible multivariate outliers can be detected by using the Mahalanobis distances via the *Outlier detection* tab as seen in Figure 5c. Finally, users can assess MVN in the *Multivariate analysis* tab by choosing one of the MVN tests including Mardia's, Henze-Zirkler's, and Royston's and graphical approaches, including chi-square Q-Q, perspective, and contour plots (Figure 5d). All the results can be downloaded by using download buttons in the respective tabs.

Summary and further research

As stated earlier, MVN is among the most crucial assumptions for most parametric multivariate statistical procedures. The power of these procedures is negatively affected if this assumption is not satisfied. Thus, before using any of the parametric multivariate statistical methods, MVN assumption should be tested first. Although there are many MVN tests, there is not a standard test for assessing this assumption. In our experience, researchers may choose Royston's test for data with a small sample size ($n < 50$) and Henze-Zirkler's test for a large sample size ($n > 100$). However, a more comprehensive simulation study is needed to provide more reliable inference. Instead of using just one test, it is suggested that using several tests simultaneously and examining some graphical representation of the data may be more appropriate. Currently, as far as we know, there is no such extensive tool to apply different statistical tests and graphical methods together.

In this paper, we present the MVN package for multivariate normality checking. This package offers comprehensive flexibility for assessing MVN assumption. It contains the three most widely used MVN tests, including Mardia's, Henze-Zirkler's and Royston's. Moreover, researchers can create three MVN plots using this package, including the chi-square Q-Q plot for any data set and perspective and contour plots for bivariate data sets. Furthermore, since MVN requires univariate normality of each variable, users can check univariate normality assumption by using both univariate normality tests and plots with proper functions in the package. In the first example, different results on multivariate normality were achieved from the same data. When *sepal* and *petal* measures, i.e., four variables, were considered, Mardia's test resulted in multivariate normality as well as Henze-Zirkler's test at the edge of type I error. However, Royston's test strongly rejected the null hypothesis in favor of non-normality. At this point, the only possible graphical approach is to use the chi-square Q-Q plot since there are more than two variables. The next step was to identify the cause of deviation from MVN by using univariate normality tests and plots. In the second example, all tests suggested bivariate normality, as did the graphical approaches. Although some tests may not reject the null hypothesis, other tests may reject it. Hence, as stated earlier, selecting the appropriate MVN test dramatically changes the results and the final decision is ultimately the researcher's.

Currently, MVN works with several statistical tests and graphical approaches. It will continue to add new statistical approaches as they are developed. The package and the web-tool will be regularly updated based on these changes.

Acknowledgments

We would like to thank Prof. Dr. C. Reha Alpar and the anonymous reviewer for their very useful comments and suggestions which helped us to improve the quality of our paper. We would also like to thank Izzet Parug Duru from Marmara University Department of Physics and Vahap Eldem from Istanbul University Department of Biology for making the web-tool version of the package possible. This study was supported by the Research Fund of Marmara University [FEN-C-DRP-120613-0273].

Bibliography

- R. Alpar. *Uygulamalı Çok Değişkenli İstatistiksel Yöntemler*. Detay Yayıncılık, Ankara, Turkey, fourth edition, 2013. ISBN 978-605-5437-42-8. [p152]
- E. Anderson. The species problem in Iris. *Missouri Botanical Garden Press*, 23(3):457–509, 1936. [p154]
- T. Burdenski. Evaluating univariate, bivariate, and multivariate normality using graphical and statistical procedures. *Multiple Linear Regression Viewpoints*, 26(2):15–28, 2000. [p151, 156]
- P. Filzmoser, R. G. Garrett, and C. Reimann. Multivariate outlier detection in exploration geochemistry. *Computers & Geosciences*, 31(5):579–587, 2005. [p159]
- R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2): 179–188, 1936. [p154]
- D. George and P. Mallery. *SPSS for Windows step by step*. Allyn & Bacon, Boston, 1999. [p151]
- I. González and S. Déjean. *CCA: Canonical correlation analysis*, 2012. URL <http://CRAN.R-project.org/package=CCA>. R package version 1.2. [p151]
- N. Henze and T. Wagner. A new approach to the BHEP tests for multivariate normality. *Journal of Multivariate Analysis*, 62(1):1–23, 1997. [p152]
- N. Henze and B. Zirkler. A class of invariant consistent tests for multivariate normality. *Communications in Statistics - Theory and Methods*, 19(10):3595–3617, 1990. [p152]
- H. Holgersson. A graphical method for assessing multivariate normality. *Computational Statistics*, 21(1):141–149, 2006. [p151]
- F. Husson, J. Josse, S. Le, and J. Mazet. *FactoMineR: Multivariate Exploratory Data Analysis and Data Mining with R*, 2014. URL <http://CRAN.R-project.org/package=FactoMineR>. R package version 1.26. [p151]
- R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, New Jersey, third edition, 1992. [p152, 153]
- R. E. Kass, U. T. Eden, and E. N. Brown. *Analysis of Neural Data*. Springer, 2014. [p156]
- S. Korkmaz, D. Goksuluk, and G. Zararsiz. *MVN: Multivariate Normality Tests*, 2014. URL <http://www.biosoft.hacettepe.edu.tr/MVN/>. R package version 3.7. [p151]
- K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3): 519–530, 1970. [p152]
- K. V. Mardia. Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies. *Sankhyā: The Indian Journal of Statistics, Series B (1960–2002)*, 36(2):115–128, 1974. [p152]
- C. J. Mecklin and D. J. Mundfrom. On using asymptotic critical values in testing for multivariate normality. *InterStat*, 1:1–12, 2003. [p152]
- C. J. Mecklin and D. J. Mundfrom. A monte carlo comparison of the type I and type II error rates of tests of multivariate normality. *Journal of Statistical Computation and Simulation*, 75(2):93–107, 2005. [p151, 153]

- S. Ramzan, F. M. Zahid, and S. Ramzan. Evaluating multivariate normality: A graphical approach. *Middle East Journal of Scientific Research*, 13(2):254–263, 2013. [p151]
- W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2014. URL <http://CRAN.R-project.org/package=psych>. R package version 1.4.8. [p151]
- G. J. S. Ross. MLP, Maximum Likelihood Program. *Harpenden: Rothamsted Experimental Station*, 1980. [p153]
- P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987. ISBN 0-471-85233-3. [p158]
- J. P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, 31(2):115–124, 1982. [p153]
- J. P. Royston. Some techniques for assessing multivariate normality based on the Shapiro-Wilk W. *Applied Statistics*, 32(2):121–133, 1983. [p153]
- P. Royston. Approximating the Shapiro-Wilk W test for non-normality. *Statistics and Computing*, 2(3): 117–119, 1992. [p153]
- P. Royston. Remark as r94: A remark on algorithm AS 181: The W test for normality. *Applied Statistics*, 44(4):547–551, 1995. [p153]
- RStudio, Inc. *shiny: Web Application Framework for R*, 2014. URL <http://CRAN.R-project.org/package=shiny>. R package version 0.10.1. [p159]
- S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1964. [p153]
- J. P. Stevens. *Applied multivariate statistics for the social sciences*. Routledge, 2012. [p156]
- T. Svantesson and J. W. Wallace. Tests for assessing multivariate normality and the covariance structure of mimo data. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 4, pages 656–659. IEEE, 2003. [p151]
- W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL <http://www.stats.ox.ac.uk/pub/MASS4>. ISBN 0-387-95457-0. [p151]

Selcuk Korkmaz
Department of Biostatistics
Hacettepe University
Turkey
selcuk.korkmaz@hacettepe.edu.tr

Dincer Goksuluk
Department of Biostatistics
Hacettepe University
Turkey
dincer.goksuluk@hacettepe.edu.tr

Gokmen Zararsiz
Department of Biostatistics
Hacettepe University
Turkey
gokmen.zararsiz@hacettepe.edu.tr

qmethod: A Package to Explore Human Perspectives Using Q Methodology

by Aiora Zabala

Abstract Q is a methodology to explore the distinct subjective perspectives that exist within a group. It is used increasingly across disciplines. The methodology is semi-qualitative and the data are analysed using data reduction methods to discern the existing patterns of thought. This package is the first to perform Q analysis in R, and it provides many advantages to the existing software: namely, it is fully cross-platform, the algorithms can be transparently examined, it provides results in a clearly structured and tabulated form ready for further exploration and modelling, it produces a graphical summary of the results, and it generates a more concise report of the distinguishing and consensus statements. This paper introduces the methodology and explains how to use the package, its advantages as well as its limitations. I illustrate the main functions with a dataset on value patterns about democracy.

Introduction

Identifying the different perspectives on or attitudes towards topics of public concern is an important research objective in fields spanning social (e.g., Zografos, 2007), environmental (e.g., Sandbrook et al., 2011) and health sciences (e.g., Thompson et al., 2001). Q is a clearly structured, systematic, and increasingly-used methodology designed specifically for these purposes (Watts and Stenner, 2012; Barry and Proops, 1999). It is aimed at exploring the distinct perspectives, discourses, or decision-making styles within a group in order to address practical matters such as the acceptance of new policies and technology or increasing public participation. The method can be used, for example, to identify student learning styles, farmer attitudes towards natural conservation (e.g., Davies and Hodge, 2012; Brodt et al., 2006), user views on technology innovation (Petit dit Dariel et al., 2013), transportation habits (van Exel et al., 2011), citizen identities (Davis, 1999), heterogeneous concepts of love (Watts and Stenner, 2005), or leadership styles in business.

In essence, the data collected in Q methodology (also known as *Q technique* or *Q-sort*) consist of a set of items (usually statements) sorted in a specific arrangement. These statements represent all possible opinions, which each respondent sorts in order to express their views (usually from *most agree* to *most disagree*).¹ The analytical process reduces the data based on principal components analysis (PCA) or factor analysis (FA). However, instead of correlating variables (as in regular PCA and FA), in Q the respondents are correlated in order to elucidate the relationships between them. The standard data reduction method is followed by a set of analytical steps specific to Q methodology. The final results consist of a small number of sets of sorted statements (typically called the *factors*), which are different from each other and summarise the perspectives existing among the respondents. These results can be used for further research: to model the relation between perspectives and other variables, to develop a quick test to identify perspectives in larger populations, or to understand the evolution of perspectives over time.

The analysis for Q methodology requires multiple matrix algebra operations which have been described in detail (see Chapter 4 and Appendix in Brown, 1980). The full analysis is implemented in software specific to Q, predominantly PQMethod, which is freely available, written in Fortran, and fully functional in Windows and Mac-OS (Schmolck, 2014). Other software include PCQ (paid-license, Windows only, Stricklin and Almeida, 2004) and Q-Assessor (paid-license, web-based, Epimetrics Group, LLC, 2010). The latter two provide tools for data collection, but the final output and report are virtually the same in all three.

This R package improves the existing Q software in a number of ways. It is fully cross-platform. It allows a completely transparent examination of the equations and the constants used in the computation at each step of the analysis, helping researchers to better understand the process. For the data reduction technique, correlation coefficients other than Pearson are also allowed. The output is concisely structured and tabulated in numerical form rather than in a large fixed width text file, therefore it provides a more straightforward and flexible means to study and handle the results. Thus **qmethod** (Zabala, 2014) results can be easily used for further quantitative modelling and for graphical representation. In addition, the final output in this package provides a clearer and more synthetic report on distinguishing and consensus statements (see below). The package also includes a specific `plot()` method to build a novel visualisation of the results, as well as import and export functionality.

¹While the Likert scales predominate in practice, Q-sorts are also widely used and include standardised sets of statements. The advantages and disadvantages are reviewed in Serfass and Sherman (2013).

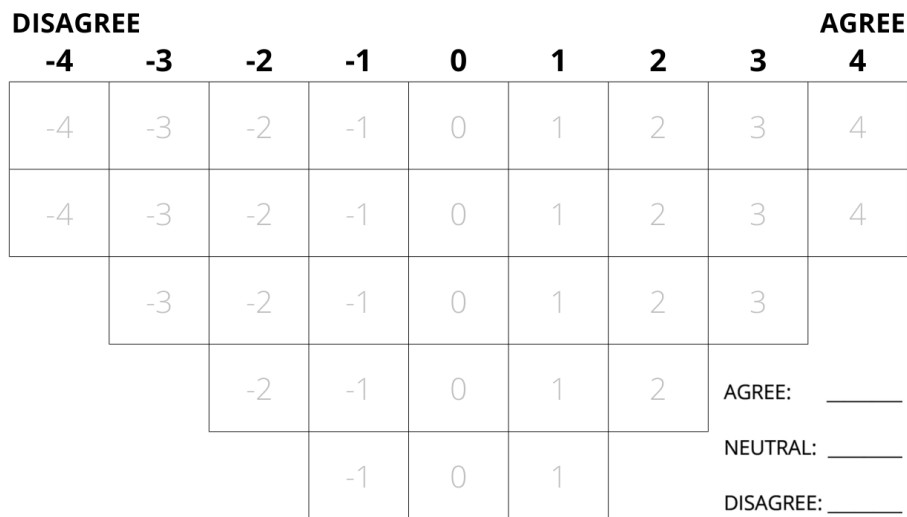


Figure 1: An example of a grid to sort statements in Q method.

The Q methodology

Q is a powerful methodology to shed light on complex problems in which human subjectivity is involved. Subjectivity is understood as how people conceive and communicate their point of view about a subject (McKeown and Thomas, 2013). The method originated from a 1935 proposal to correlate respondents instead of variables in FA by Stephenson (Stephenson, 1935), an assistant to Spearman—the developer of FA. Q was used initially in psychology, then in political science, and, after that, in several other fields. The analytical process is clearly structured and well established (Stephenson, 1953; Brown, 1980), and Q is increasingly being used across disciplines and for different purposes such as policy evaluation, understanding decision-making, or participatory processes.

The following characteristics of the methodology will help in deciding whether it is a suitable approach for a given question. It is versatile due to its compatibility with small samples (see below). It is predominantly exploratory because the patterns of views emerge from the study and thus prevent the researcher from imposing a frame of reference or pre-determined assumptions and definitions (McKeown and Thomas, 2013; Stenner et al., 2008). It is a mixed or semi-qualitative methodology because though the data collected are quantitatively analysed, their interpretation is extensively qualitative (Ramlo, 2011) and makes thorough use of theory. The results can be used in combination with other qualitative methods and as a starting point for quantitative confirmatory methods. For example, Q can be combined with discourse analysis, or it can be used in regression models to examine how perspectives influence behaviour. The basic analytical principle is to correlate the entire responses of individuals. These responses are measured using the same unit, which is often called *psychological significance* or *self-significance*, and they indicate the salience (engagement or disengagement) of the statement for the respondent. Both aspects contrast with regular FA, in which variables are correlated and, having different units, may also be incommensurable.

Research design

In its most frequent form, the Q approach consists of selecting a set of statements and asking respondents to sort them over a grid, from *most agreement* to *most disagreement* (see Figure 1 for an example of a grid). The statements are a representative sample of the *concourse*, the whole set of possible expressions on a topic, gathered from all possible points of view (in theory, a concourse would be infinite). The researcher collects a large set of statements from interviews, reviews of literature or mass media, expert consultation, participant observation, etc. This collection is reduced to a final representative selection that usually ranges between 40 and 80 statements (Watts and Stenner, 2012). The statements can express understandings or behavioural preferences relating to the topic. Occasionally, photos, sounds, or other types of stimuli may be used instead of statements.

The sample of respondents does not need to be large or representative of the population, but it must be diverse. The aim is to get the most diverse range of opinions, regardless of whether they are minority ones. The shape of the grid used to sort the statements is up to the researcher. This grid is usually bell-shaped as in Figure 1, assuming that fewer statements generate strong engagement (Brown, 1980). Respondents commonly sort the statements according to their agreement or disagreement, although there are other possible *conditions of instruction*—different ways in which participants are

asked to sort the statements (McKeown and Thomas, 2013), e. g., “Sort the statements according to how person A would respond”. A succinct description of the research design can be found in van Exel and de Graaf (2005), Watts and Stenner (2012) offer a detailed reference manual, and a key and extensive work is that of Brown (1980).

Analytical process

The data collected from all respondents are introduced into a matrix with statements as rows and respondents as columns, where the cell values are the score in the grid in which the respondent sorted the statement. For example, in Figure 1 the statements that a respondent most disagreed with would get a score of -4 . Sample data available with the package can be loaded by using the command `data(lipset)`. The array of scores for all the statements sorted by a single respondent (the column) is called the *Q-sort*.

The process of analysis has two main parts. In the first, a multivariate data reduction technique is applied, either centroid factor analysis or PCA. This package currently implements only the PCA solution. PCA is readily available in R, and the results from both techniques are similar (McKeown and Thomas, 2013; Watts and Stenner, 2012). The centroid algorithm for factor analysis is an alternative method for FA used almost exclusively in Q methodology and described in Brown (1980). This algorithm differs from standard FA (as implemented in `factanal()`) and their results, although highly correlated, are not identical.

Initially, a correlation matrix between Q-sorts is built, and the chosen multivariate technique reduces this correlation matrix into components. The components are ordered by the total variability that they explain, and so the first components summarise most of the variability of the initial correlation matrix. Then the first few components are selected and rotated in order to obtain a clearer and simpler structure of the data. The usual criteria by which the number of components is selected include, *inter alia*, the total amount of variability explained, eigenvalues higher than a certain threshold—both accessible through the call `loadings(principal(...))` from `psych` (Revelle, 2014), and a compromised solution between complexity and interpretability (further details about the possible criteria are given in Watts and Stenner, 2012).

The rotation of components in Q studies can be either manual (judgemental) or mathematically optimal (analytical). The rotation results in a matrix of component *loadings* with Q-sorts as rows and components as columns, indicating the relationship between each Q-sort and component. Mathematical rotation is implemented in the package within the function `qmethod()`. This function calls internally `principal()` from `psych`, which conveniently wraps the rotation modes from `GPARotation` (Bernaards and Jennrich, 2005) into a single function. Any of the rotations implemented in `principal()` can be called in `qmethod`, and “`varimax`” is the most commonly used. Manual rotation is not integrated in the current version.²

The second part of the analysis is particular to Q. It consists of a) flagging the Q-sorts that will define each component (hereafter called *factor*, as it usually is in the literature; implemented in the function `qflag()`), b) calculating the scores of statements for each factor (z-scores and factor scores, implemented in `qzscore()`), and c) finding the distinguishing and consensus statements (implemented in `qda()`).³

The most representative Q-sorts for each factor are *flagged* (a), meaning that only these Q-sorts are used for subsequent calculations. The purpose of flagging is to obtain more distinguishable perspectives, and it may be done either automatically or manually, the latter occurring when the researcher has relevant knowledge about any of the respondents. Automatic flagging is based on two criteria: that the loading ℓ should be significantly high (the significance threshold for a p-value $< .05$ is given by equation 1, where N is the number of statements; Brown 1980), and that the square loading for a factor j should be higher than the sum of the square loadings for all other factors (equation 2, where f is the total number of factors; Brown 1980). Some Q-sorts may be considered confounding because they load highly in more than one factor and thus they are not flagged. Alternatively, manual flagging may be used (see details on how to run manual flagging in [Implementation of the analysis in](#)

²The graphical interface PQROT, for Windows and Mac-OS only, complements PQMethod and allows manual rotation.

³All the necessary equations are detailed in Brown (1980) and may be examined in the corresponding R function by typing its name, e. g., `qzscore`.

`qmethod`).

$$\ell > \frac{1.96}{\sqrt{N}} \quad (1)$$

$$\ell_j^2 > \sum_{i=1}^f \ell_i^2 - \ell_j^2 \quad (2)$$

The z-scores (b) indicate the relationship between statements and factors, i. e., how much each factor *agrees* with a statement. The z-score is a weighted average of the scores given by the flagged Q-sorts to that statement. The factor scores are obtained by rounding the z-scores towards the array of discrete values in the grid. In Figure 1, this array of discrete values would be $c(-4, -4, -3, -3, -3, -2, \dots, 4)$.⁴ The final outcome of the analysis is the selected number of factors, representing one perspective each. These perspectives are a hypothetical Q-sort that has been reconstructed from the factor scores.

Next, some general characteristics are calculated in order to compare the factors. For each factor, the following are provided: the number of flagged Q-sorts, the composite reliability, and the standard error (*SE*) of factor scores. Two additional matrices indicate the similarity between the z-scores of each pair of factors: the correlation coefficients and the standard error of differences (*SED*, based on the *SE*).⁵

Finally, the factor comparison identifies the consensus and distinguishing statements (c). For each pair of factors, if the difference between the z-scores of a statement is statistically significant (based on the *SED*), then what both factors *think* about that statement is distinct. When none of the differences between any pair of factors are significant, then the statement is considered of consensus.

Interpretation and reporting

The interpretation of each perspective is based on the Q-sort reconstructed from the factor scores and on the salience and distinctiveness of the statements. Each respondent may be more closely related to one of the perspectives, and this relation is determined by the loadings calculated at the beginning. The key elements to look at are the relative position of statements within the grid (particularly those at the extremes), the position of a statement in a perspective versus the position of the same statement in other perspectives, and the distinguishing and consensus statements. Each perspective is given a semantic denomination and is described in as much length as necessary, each description deriving from the literature and from qualitative explanations collected after each response.

The essential characteristics of a Q study include the process of selecting statements, the shape of the distribution grid, the number of participants and the criteria for their selection, the methods for extraction and rotation of factors, and the number of Q-sorts loading on each factor. The results are usually reported with a table of statements including either their z-scores or factor scores, and an indication of which statements are distinguishing and which consensus. The table of factor loadings may also be included, showing the Q-sorts that were flagged.

Implementation of the analysis in `qmethod`

The core of the package consists of a main function `qmethod()` and four subordinate functions that conform to the steps of the analysis: `qflag()`, `qzscores()`, `qfcharacter()`, and `qdc()`. The function `qmethod()` is a wrapper that calls internally PCA to calculate loadings and the four other functions. The individual functions can be run independently to build the analysis step-by-step in order to maintain more control over what happens at each stage or to perform more advanced analysis. Yet running the individual steps will rarely be necessary unless the researcher wants to use other methods for extraction or manual flagging. The core functionality is complemented with additional functions to print, summarize, plot, import, and export.

The raw data is provided to `qmethod()` as a matrix or data frame with statements as rows and Q-sorts as columns. The number of factors to extract is necessary, and this can be decided upon exploration of the raw data based on criteria recommended in the literature, as explained above in [Analytical process](#). The method for rotation is "varimax" by default, but other methods can be specified. If respondents do not have to follow the distribution grid strictly when sorting the statements, then the argument `forced` should be `FALSE` and a vector must be provided in the argument

⁴The calculations of both z-scores and rounded scores are implemented in the function `qzscores()`. The function calculates factor weights and the weighted average scores for each statement. Then it standardizes the scores into the z-scores and rounds the z-scores to the discrete values of the distribution.

⁵All the general factor characteristics are calculated in the function `qdcharacter()`. These are based on the loadings, the flagged Q-sorts, and a matrix of z-scores resulting from `qzscores()`.

distribution. This distribution vector is the array of values corresponding to the grid. By calling `qmethod()` with all the necessary arguments, the full analysis is performed and the outputs are put together in an object of class "QmethodRes".

In order to run manual flagging, the functions corresponding to individual steps may be used instead of `qmethod()`: namely, `qzscores()` and `qdc()` (`qfcharacter()` is called internally in `qzscores()`). First, and in order to assess which Q-sorts to flag, one may run the function `qflag()` and examine the resulting table of loadings. Second, in `qzscores()` a logical matrix of n Q-sorts and f factors may be provided in the argument `flagged`, where the cells may be TRUE to indicate flagging. After calculating the z-scores, distinguishing and consensus statements may be identified using the function `qdc()`.

The package also allows the use of correlation coefficients other than Pearson for the extraction of factors, namely Spearman and Kendall. These may be appropriate for non-parametric data and may sometimes enable a greater amount of variability to be explained with fewer factors (for a technical note about correlation coefficients, see [Brown, 1980](#), p. 276).

Understanding and exploring results from the `qmethod()` function

The function `qmethod()` returns the results in a list of class "QmethodRes" containing eight objects. The method `print()` for an object of class "QmethodRes" provides a snapshot of the full results with descriptive names for each object within the list, as listed below (in parenthesis, the actual names of the objects within the list).⁶ The method `summary()` displays the essential tables. In order to visualize the results at a glance, the method `plot()` builds a dot-chart of z-scores, as in [Figure 2](#).

1. "Q-method analysis" (`...$brief`): a list with basic information of the analysis including date, number of Q-sorts and of statements, number of factors extracted, and rotation.
2. "Original data" (`...$dataset`): a data frame with the raw data.
3. "Q-sort factor loadings" (`...$loa`): a data frame with the rotated loadings obtained from `principal()`.
4. "Flagged Q-sorts" (`...$flagged`): a logical data frame indicating which Q-sorts are flagged for which factors, obtained from `qflag()`.
5. "Statement z-scores" (`...$zsc`): the weighted average value of each statement for each factor, obtained from `qzscores()`.
6. "Statement factor scores" (`...$zsc_n`): the scores rounded to match the array of discrete values in the distribution, obtained from `qzscores()`.
7. "Factor characteristics" (`...$f_char`): a list of three objects, obtained from `qfcharacter()`:
 - (a) A matrix with the general characteristics of each factor (`...f_charcharacteristics`):
 - Average reliability coefficient
 - Number of loading Q-sorts
 - Eigenvalues
 - Percentage of explained variance
 - Composite reliability
 - Standard error of factor scores
 - (b) The matrix of "Correlation between factor z-scores" (`...f_charcor_zsc`).
 - (c) The matrix of "Standard errors of differences between factors" (`...f_charsd_dif`).
8. "Distinguishing and consensus statements" (`...$qdc`): a data frame that compares the z-scores between all pairs of factors, obtained from `qdc()`.

The last object "Distinguishing and consensus statements" may be explained in detail. This object results from an internal call to the function `qdc()`. For each pair of factors, this function calculates the absolute difference in z-scores and compares this difference with the significance thresholds for .05 and .01 p-value levels. The function `qdc()` returns a data frame with statements as rows and comparisons as columns. All the comparisons are synthesised in the first variable of the data frame, which is a categorical variable named "dist.and.cons" that indicates whether the statement is of consensus or distinguishing for one or more factors (see an example below in [Usage example](#)). The following are the possible categories that a statement can fall into in the "dist.and.cons" variable:

- "Distinguishes all": When all the differences between all pairs of factors are significant.

⁶Note that in the output of results the statements have a unique ID of the type "n*" and the actual text of the statements is not merged. Adding the corresponding text of the statements to the final results can be easily done using, for example, `merge()`, `cbind()` or `rownames()`. See an example below in [Usage example](#).

- “Distinguishes f_i only”: When the differences between factor i and all other factors are significant, and the differences between all other pairs of factors are not.
- “Distinguishes f_i (...)”: When the differences between factor i and all other factors are significant, and some (but not all) of the differences between other pairs of factors are significant. If this is the case for more than one factor, the string is concatenated, e. g., “Distinguishes f_1 Distinguishes f_3 ”. This category may arise only in solutions of four or more factors.
- “Consensus”: When none of the differences are significant because all factors give the statement a similar score.
- “”: Leaves an empty string in the cell of those statements which do not fulfil any of the above conditions, i. e., statements that are neither consensus nor clearly distinguishing any factor from all the rest. But while they do not distinguish any particular factor from all the rest, they do distinguish some pairs of factors. The role of these statements may be inspected in detail by looking at the significance columns.

This structure of results is different from that of other Q software and it contains all the necessary information without any redundancy. This output can be converted into the exact outline provided by PQMethod by using the function `export.qm()` (see below), an outline that is much longer. Most of this conversion consists of taking the data frames of z-scores, of factor scores, and of distinguishing and consensus statements (objects 5, 6, and 8 within the list of results), and reordering or merging them according to different criteria.

Importing and exporting data

The function `import.pqmethod()` retrieves data from a .DAT file, which is the raw data file saved by PQMethod software. Individual data frames from a “QmethodRes” object may be exported as a CSV using, for example, `write.table()` (to find the objects to export from within the list of results, see the description of the outputs above in [Understanding and exploring results from the `qmethod\(\)` function](#)). The function `export.qm()` saves all the results obtained from `qmethod()` in a text file, building the report which is then used for the interpretation. This report has two flavours defined in the argument `style`: “R” and “PQMethod”. “R” exports the results exactly as the function `qmethod()` returns them. “PQMethod” exports the results following the structure of the output in PQMethod (a .LIS file). Note that the latter is a much longer outline and has some redundant information in the form of tables reordered according to different criteria. This alternative outline might be convenient for researchers accustomed to PQMethod.

Usage example

For demonstration purposes, I use the *Lipset* dataset about the value patterns of democracy ([Lipset 1963](#) and [Stephenson 1970](#), both in [Brown, 1980](#)), which contains 9 respondents and 33 statements. The following code performs a full analysis using principal components and varimax rotation to extract three components (*factors*).

```
data(lipset)
lipset[[1]] # Shows the dataset, a matrix of 33x9
lipset[[2]] # Shows the text of the 33 statements
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
```

The object `results` is of class “QmethodRes”, and the specific method `summary()` for this class returns the basic information and the data frame of factor scores as shown below. This data frame contains the three factors or main perspectives. Each perspective has a distinct array of statement scores, which correspond to the scores in [Figure 1](#) and indicate the agreement or disagreement of the given perspective with each statement. For example, perspective one is in strong agreement with statement 1 (“sta_1” has a score of 4), whereas the statement deserves the opposite opinion according to perspective two (a score of -2) and perspective three considers it in the middle ground (a score of 1). The next matrix contains general information about each factor, of which the most relevant piece may be the number of loading Q-sorts and the explained variance, which are approximate indicators of the strength of each perspective and of the proportion of the opinions they explain.

```
> summary(results)
Q-method analysis.
Finished on:      Tue Oct 21 10:22:50 2014
Original data:   33 statements, 9 Q-sorts
Number of factors: 3
```

```

Rotation:          varimax
Flagging:         automatic
Correlation coefficient: pearson

```

Factor scores

	fsc_f1	fsc_f2	fsc_f3
sta_1	4	-2	1
sta_2	0	1	-3
sta_3	-3	-1	-1
sta_4	2	-3	2
sta_5	-1	-1	3
sta_6	0	3	3
sta_7	-4	1	-2
sta_8	-3	0	-1
sta_9	2	-3	-1
sta_10	-4	-1	-2
sta_11	-2	2	2
sta_12	1	0	-1
sta_13	3	3	1
sta_14	-2	0	0
sta_15	-1	2	-4
sta_16	-3	-4	4
sta_17	0	-1	0
sta_18	1	-2	1
sta_19	3	-2	1
sta_20	-1	-1	0
sta_21	2	4	-3
sta_22	-2	0	-2
sta_23	0	2	-1
sta_24	2	1	-4
sta_25	1	1	2
sta_26	3	1	1
sta_27	-2	2	0
sta_28	0	3	4
sta_29	-1	0	-2
sta_30	1	-4	2
sta_31	-1	-2	0
sta_32	4	-3	3
sta_33	1	4	-3

	f1	f2	f3
Average reliability coefficient	0.80	0.80	0.80
Number of loading Q-sorts	3.00	3.00	3.00
Eigenvalues	2.09	1.97	1.68
Percentage of explained variance	23.17	21.93	18.68
Composite reliability	0.92	0.92	0.92
Standard error of factor scores	0.28	0.28	0.28

Any of the results may be retrieved by using the corresponding object name indicated under [Understanding and exploring results from the `qmethod\(\)` function](#), and thus customised for easier exploration. For instance, the z-scores may be shown by using the command `results$zsc`. In the example below, the factor scores are merged with the actual text of the statements and then ordered. The data frame is reordered according to the scores of the statements for each factor, so that the researcher can quickly identify which statements are in most agreement for a given perspective, and what other perspectives think of the same statements:

```

# Merge the statements with their actual text:
scores <- cbind(results$zsc_n, lipset[[2]])

# Order the results by the scores of each factor:
for (i in 1:length(results$loa)) {
  View(scores[order(scores[i], decreasing = TRUE), ],
        title = paste0("Order for f", i))
}

```

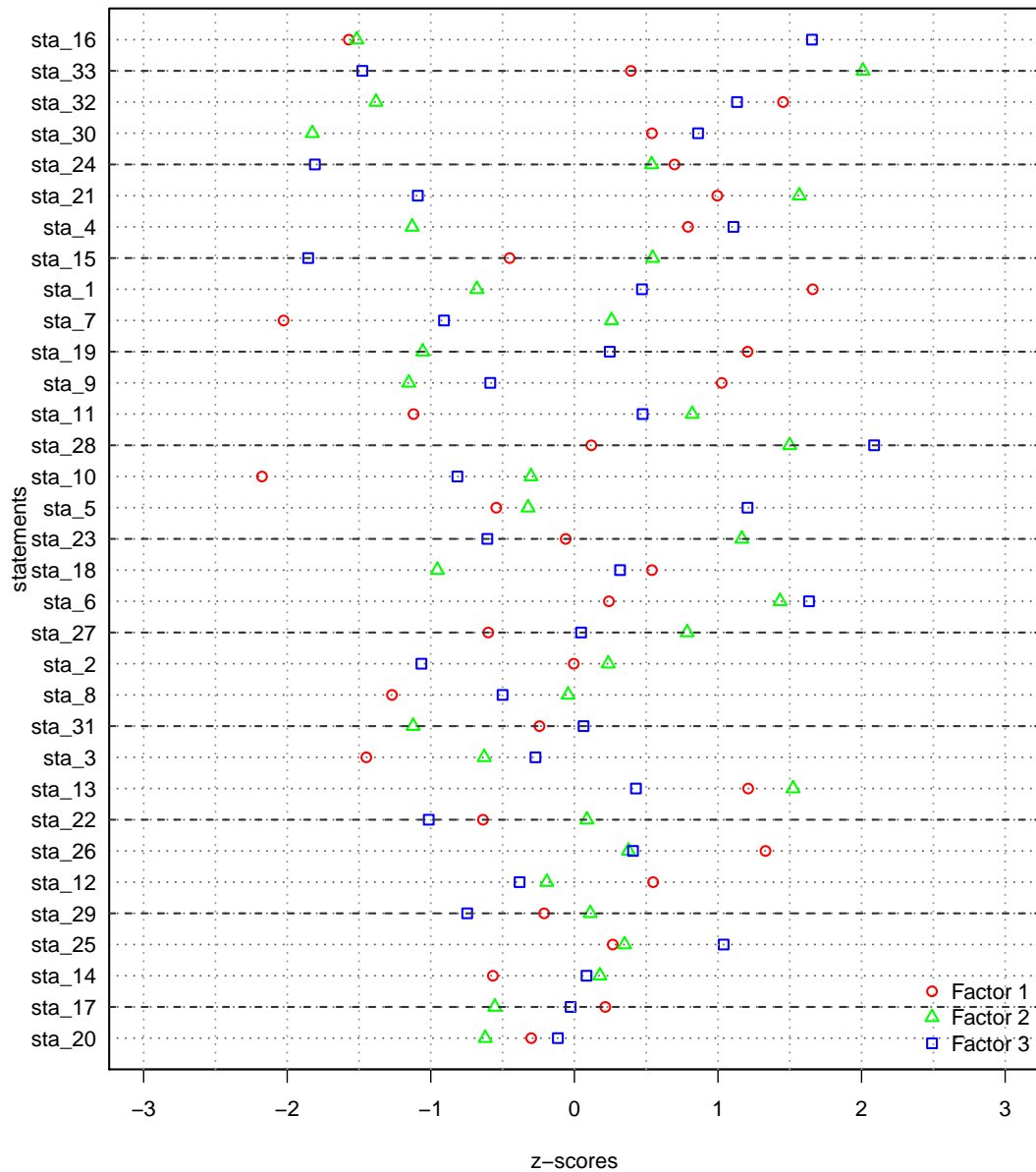


Figure 2: An example of a plot of an object of class "QmethodRes". The statements are ordered by the standard deviation of z-scores for all three factors.

The method `plot()` for class "QmethodRes" returns a dot-chart of the z-scores specifically adapted for Q methodology, as in Figure 2. In this figure, built with the code below, the comparison among the z-scores of all factors can be explored. For example, all three points are far from each other in statement 33, meaning that each of the three factors holds a distinctive opinion regarding this statement. For statement 20, however, the points are clustered together, indicating consensus. Finally, statement 16 clearly distinguishes factor three from the rest (its point being far from the other two).

```
par(lwd = 1.5, mar = c(4, 4, 0, 0) + 0.1)
plot(results)
abline(h = seq(from = 2, to = 32, by = 3), col = grey(0.2), lty = 2)
```

The table of distinguishing and consensus statements below conveys the observations gleaned from Figure 2 with greater precision. For example, column "f1_f2" shows the absolute difference in z-scores between factor one and factor two. In the column immediately to the right ("sig_f1_f2"), a single star or double star indicate differences that are significant at p-values < .05 and < .01 respectively, and arise from the magnitude of the difference and the thresholds given by the SED.

```

> # Data frame of distinguishing and consensus statements:
> format(results$qdc, digits = 1, nsmall = 2)
      dist.and.cons f1_f2 sig_f1_f2 f1_f3 sig_f1_f3 f2_f3 sig_f2_f3
sta_1 Distinguishes all 2.34      ** 1.19      ** 1.15      **
sta_2 Distinguishes f3 only 0.24      1.06      ** 1.30      **
sta_3 Distinguishes f1 only 0.82      * 1.18      ** 0.36
sta_4 Distinguishes f2 only 1.92      ** 0.32      2.24      **
sta_5 Distinguishes f3 only 0.22      1.75      ** 1.53      **
sta_6 Distinguishes f1 only 1.19      ** 1.39      ** 0.20
sta_7 Distinguishes all 2.28      ** 1.12      ** 1.17      **
sta_8 Distinguishes f1 only 1.23      ** 0.77      * 0.46
sta_9 Distinguishes f1 only 2.18      ** 1.61      ** 0.57
sta_10 Distinguishes f1 only 1.87      ** 1.36      ** 0.51
sta_11 Distinguishes f1 only 1.94      ** 1.60      ** 0.35
sta_12      0.74      0.93      * 0.19
sta_13 Distinguishes f3 only 0.31      0.78      * 1.09      **
sta_14      Consensus 0.75      0.65      0.09
sta_15 Distinguishes all 1.00      * 1.40      ** 2.40      **
sta_16 Distinguishes f3 only 0.06      3.23      ** 3.17      **
sta_17      0.77      * 0.24      0.53
sta_18 Distinguishes f2 only 1.49      ** 0.22      1.27      **
sta_19 Distinguishes all 2.26      ** 0.96      * 1.30      **
sta_20      Consensus 0.32      0.19      0.51
sta_21 Distinguishes f3 only 0.57      2.09      ** 2.66      **
sta_22      0.72      0.38      1.10      **
sta_23 Distinguishes f2 only 1.23      ** 0.55      1.77      **
sta_24 Distinguishes f3 only 0.16      2.50      ** 2.35      **
sta_25      0.08      0.77      * 0.69
sta_26 Distinguishes f1 only 0.95      * 0.92      * 0.03
sta_27      1.39      ** 0.65      0.74
sta_28 Distinguishes f1 only 1.38      ** 1.97      ** 0.59
sta_29      0.32      0.54      0.86      *
sta_30 Distinguishes f2 only 2.36      ** 0.32      2.69      **
sta_31 Distinguishes f2 only 0.88      * 0.31      1.19      **
sta_32 Distinguishes f2 only 2.83      ** 0.32      2.51      **
sta_33 Distinguishes all 1.62      ** 1.87      ** 3.49      **

```

In the above example, the statements 3, 6, 8, 9, etc. (labelled "Distinguishes f1 only") distinguish factor one (f_1) but do not distinguish f_2 from f_3 . The statements 1, 7, 15, 19, and 33 (labelled "Distinguishes all") distinguish both f_1 from the other two and also f_2 from f_3 : all factors think differently about these statements. Meanwhile, statements 14 and 20 are of consensus because none of their differences are significant at p -level = .05 (no stars appear in any of the "sig_*" columns). In addition, those statements with empty values under "dist.and.cons" need to be looked at individually (statements 12, 17, 22, 25, 27, and 29). For example, statements 12 and 25 distinguish f_1 from f_3 , but they do not distinguish either against f_2 (whose p -value is $< .05$ as indicated in column "sig_f1_f3", but none of the other comparisons are significant).

Validation

The package was validated with the `lipset` dataset and with three other datasets, extracting 2, 3, 4 and 5 factors with each of them. The results of `qmethod` were contrasted with the results of analyses based on the same options but performed in `PQMethod`. For studies of 1 to 3 factors, all the numbers in factor loadings and z-scores match to the fourth decimal those given in `PQMethod`. For studies of 4 or more factors, all the numbers match to the second decimal. Occasional divergences in the third and fourth decimals of the loading values arise from the PCA algorithms themselves, which are coded externally to this package.⁷ The factor scores match in all cases.

The selection of distinguishing statements matches exactly. A difference in the selection of consensus statements is due to a greater restrictiveness in this package. For in `qmethod`, the only statements identified as consensus are those in which none of the differences are significant at p -value $< .05$

⁷Four different R functions for principal components were compared with two datasets to test whether the method employed is the source of this divergence: `prcomp()`, `princomp()`, both from the base package `stats`, `principal()` from `psych`, and `PCA()` from `FactoMineR` (Husson et al., 2014). The results show that loadings and z-scores obtained from these methods differ only after the 14th decimal.

(that is, only those statements which do not produce stars in any columns). PQMethod also indicates consensus statements with no significances at p -value $< .05$ with a star, but it further identifies as *consensus* those statements with some differences significant at a p -value between $.01$ and $.05$ (these statements have single stars in some of the comparisons, though no double stars). In PQMethod, therefore, the statements with differences significant at a p -value between $.01$ and $.05$ are shown both consensus and the distinguishing lists for some or all of the factors. For example in the above table, statements 12, 17, 25, 26, and 29 have no double stars but have one or more single stars; in PQMethod these would be included as both *distinguishing* and *consensus* statements. Such double labelling can be confusing in the interpretation. Whereas in this package the statements with differences significant at a p -value between $.05$ and $.01$ are not labelled as consensus, but rather as "Distinguishes f*", "Distinguishes all", or "", depending on each case. The role of each statement can be fully understood by inspecting the table of distinguishing and consensus statements.

The order of factors in the matrices (e. g., in the matrix of loadings) may differ between both tools in some cases. This is because in R, the components in PCA are ordered according to the explained variance of the rotated components. In PQMethod, the factors are ordered according to the explained variance of the unrotated factors instead. This discrepancy affects neither the numerical results nor the interpretation.

Summary and future work

Q is an effective methodology for understanding the diversity of perspectives across disciplines. **qmethod** is the first R package to analyse Q methodology data. This package produces tabulated results that are easy to examine and interpret, and ready for graphical representation or further numerical analysis. It provides a more concise output of distinguishing and consensus statements as well as a synthesising plot function. This core functionality is complemented by additional functions that import data from other Q software, summarise the results, and export the outputs in plain text for the interpretation in two flavours. Further usage details can be found in the **qmethod** reference manual available from CRAN. Potential developments for the current implementation include the introduction of centroid extraction as an alternative to PCA, manual rotation of factors, a graphical interface, functions for data collection, and a 3D plot method to explore the results further. Researchers who would like to contribute to these or other developments are welcome to contact the author.

Acknowledgements

During the development of this package, the author was funded by the Department of Research of the Basque Government. The author is grateful to Steven Brown and Peter Schmolck for making Q datasets publicly available and for allowing the *Lipset* dataset to be used in this R package; to Laurent Gatto for his advice in developing it; and to Ben Fried, two anonymous reviewers, and the editor for their useful comments on the manuscript.

Bibliography

- J. Barry and J. Proops. Seeking sustainability discourses with Q methodology. *Ecological Economics*, 28(3):337–345, 1999. [p163]
- C. A. Bernaards and R. I. Jennrich. Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis. *Educational and Psychological Measurement*, 65:676–696, 2005. [p165]
- S. Brodt, K. Klonsky, and L. Tourte. Farmer goals and management styles: Implications for advancing biologically based agriculture. *Agricultural Systems*, 89(1):90–105, 2006. [p163]
- S. R. Brown. *Political Subjectivity: Applications of Q Methodology in Political Science*. Yale University Press, New Haven and London, 1980. URL <http://qmethod.org/papers/Brown-1980-PoliticalSubjectivity.pdf>. [p163, 164, 165, 167, 168]
- B. B. Davies and I. D. Hodge. Shifting environmental perspectives in agriculture: Repeated Q analysis and the stability of preference structures. *Ecological Economics*, 83:51–57, 2012. [p163]
- T. C. Davis. Revisiting group attachment: Ethnic and national identity. *Political Psychology*, 20(1):25–47, 1999. [p163]
- Epimetrics Group, LLC. Q-Assessor, 2010. URL <http://q-assessor.com/>. [p163]

- F. Husson, J. Josse, S. Le, and J. Mazet. *FactoMineR: Multivariate Exploratory Data Analysis and Data Mining with R*, 2014. URL <http://CRAN.R-project.org/package=FactoMineR>. R package version 1.27. [p171]
- S. Lipset. The value patterns of democracy: A case study in comparative analysis. *American Sociological Review*, 28(4):515–531, 1963. [p168]
- B. McKeown and D. Thomas. *Q-Methodology*. Sage Publications, London, 2013. [p164, 165]
- O. Petit dit Dariel, H. Wharrad, and R. Windle. Exploring the underlying factors influencing e-learning adoption in nurse education. *Journal of Advanced Nursing*, 69(6):1289–1300, 2013. [p163]
- S. E. Ramlo. Q methodology and its position in the mixed-methods continuum. *Operant Subjectivity*, 34(3), 2011. [p164]
- W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2014. URL <http://CRAN.R-project.org/package=psych>. R package version 1.4.4. [p165]
- C. Sandbrook, I. R. Scales, B. Vira, and W. M. Adams. Value plurality among conservation professionals. *Conservation Biology*, 25(2):285–94, 2011. [p163]
- P. Schmolck. PQMethod (version 2.35), 2014. URL <http://schmolck.org/qmethod/>. [p163]
- D. G. Serfass and R. A. Sherman. A methodological note on ordered Q-Sort ratings. *Journal of Research in Personality*, 47(6):853–858, 2013. [p163]
- P. Stenner, S. Watts, and M. Worrell. Q Methodology. In C. Willig and W. Stainton-Rogers, editors, *The SAGE Handbook of Qualitative Research in Psychology*, chapter 13, pages 215–239. Sage Publications, London, 2008. [p164]
- W. Stephenson. Technique of factor analysis. *Nature*, 136:297, 1935. [p164]
- W. Stephenson. *The Study of Behavior; Q-Technique and its Methodology*. University of Chicago Press, Chicago, 1953. [p164]
- W. Stephenson. *The Value Patterns of Democracy*. University of Missouri, Columbia, 1970. [p168]
- M. Stricklin and R. Almeida. PCQ (version 1.4.1), 2004. URL <http://www.pcqsoft.com/>. [p163]
- C. Thompson, D. McCaughan, N. Cullum, T. A. Sheldon, A. Mulhall, and D. R. Thompson. The accessibility of research-based knowledge for nurses in United Kingdom acute care settings. *Journal of Advanced Nursing*, 36(1):11–22, 2001. [p163]
- N. J. A. van Exel and G. de Graaf. *Q Methodology: A Sneak Preview*. 2005. URL <http://qmethod.org/articles/vanExel.pdf>. [p165]
- N. J. A. van Exel, G. de Graaf, and P. Rietveld. “I can do perfectly well without a car!”. *Transportation*, 38:383–407, 2011. [p163]
- S. Watts and P. Stenner. The subjective experience of partnership love: A Q methodological study. *The British Journal of Social Psychology*, 44(Pt 1):85–107, 2005. [p163]
- S. Watts and P. Stenner. *Doing Q Methodological Research: Theory, Method & Interpretation*. Sage Publications Ltd, London, 2012. [p163, 164, 165]
- A. Zabala. *qmethod: An R Package to Analyse Q Method Data*. University of Cambridge, Cambridge, UK, 2014. URL <http://CRAN.R-project.org/package=qmethod>. R package version 1.2.4. [p163]
- C. Zografos. Rurality discourses and the role of the social enterprise in regenerating rural Scotland. *Journal of Rural Studies*, 23(1):38–51, 2007. [p163]

Aiora Zabala
 Department of Land Economy
 University of Cambridge
 19 Silver Street, Cambridge CB3 9EP
 UK
aiora.zabala@gmail.com

gset: An R Package for Exact Sequential Test of Equivalence Hypothesis Based on Bivariate Non-Central t -Statistics

by Fang Liu

Abstract The R package `gset` calculates equivalence and futility boundaries based on the exact bivariate non-central t test statistics. It is the first R package that targets specifically at the group sequential test of equivalence hypotheses. The exact test approach adopted by `gset` neither assumes the large-sample normality of the test statistics nor ignores the contribution to the overall Type I error rate from rejecting one out of the two one-sided hypotheses under a null value. The features of `gset` include: error spending functions, computation of equivalence boundaries and futility boundaries, either binding or nonbinding, depiction of stagewise boundary plots, and operating characteristics of a given group sequential design including empirical Type I error rate, empirical power, expected sample size, and probability of stopping at an interim look due to equivalence or futility.

Introduction

Group sequential tests are repeated significance testing on data accumulated during a study, in contrast to the traditional one-time analysis at the end of the study. Since the same hypothesis is repeatedly tested, it is critical to compute the proper critical values at each interim analysis to keep the overall Type I error rate at a prespecified nominal level. Applied appropriately, group sequential designs (GSDs) can help saving resources, shortening study duration, and stopping ineffective treatments much earlier than the traditional non-sequential designs. There are existing software applications, both commercial and open-source, of GSDs in studies, including PROC SEQDESIGN and PROC SEQTEST procedures in SAS®, EAST® developed by Cytel, as well as the R packages `gsDesign` (general GSDs and operating characteristics; Anderson 2014), `GroupSeq` (GSD via the Type I error spending approach; Pahl 2014), the `ldBand` function from `Hmisc` (GSD via from the Lan-DeMets method with a variety of α -spending functions; Harrell Jr 2014), `ldbands` (boundary calculation using the Lan-DeMets α spending function approach), `PwrGSD` (evaluation of interim analysis plans for GSD on a survival endpoint; construction of efficacy and futility boundaries, and calculation of power of a GSD; Izmirlan 2014), `AGSDest` (parameter estimation in adaptive GSDs; Hack et al. 2013), `clinfun` (design and analysis of clinical trials; Seshan 2014).

This discussion focuses on GSDs in studies with equivalence hypotheses. Equivalence studies concern “equivalence” between two groups. Since mathematical equivalence is impossible to establish, the concept of “equivalence bounds” is often applied. Denote the parameter that represents the dissimilarity in a response variable between two groups by θ , then the hypothesis set being tests in equivalence studies can be expressed as two one-sided hypotheses $\{H_{10}: \theta < L \text{ v.s. } H_{11}: \theta > L\}$ and $\{H_{20}: \theta > U \text{ v.s. } H_{21}: \theta < U\}$, where equivalence bounds L and U are located on the opposite sides of the value that represents mathematical equivalence though not necessarily symmetric. The choice of (L, U) is primarily driven by practical and scientific justifications, or set by regulatory requirements in case of pharmaceutical studies. Simultaneous rejection of H_{10} and H_{20} leads to the claim of equivalence. GSDs with equivalence hypotheses have been discussed in the literature (see Jennison and Turnbull 1989; Durrleman and Simon 1990; Jennison and Turnbull 1993; Gould 1995; Whitehead 1996; Hwang 1996; Huang and Self 2010; Potvin et al. 2008; Montague et al. 2012). Many test procedures discussed in the literature are based on the large-sample normality assumption of the test statistics. This could be problematic in sequential tests given that the stagewise sample sizes at the early looks can be small. Secondly, to facilitate the computation of the critical values, rather than working with the dual test statistics (one for each of the two one-sided hypothesis), which is computationally more demanding, an analytical approximation that ignores the contribution to the probability of rejecting one H_0 out of the two under H_0 is commonly applied in practice. Though this approach is valid in the sense that it guarantees the Type I error rate at or below the nominal level, it can be overly conservative when the Type I error rate or sample size is small, which can be the very situation that GSDs have to deal with in early stages. The exact test procedure formed with the dual t statistics in the latest work of Liu and Li (2014) overcomes these major methodological shortcomings in the GSDs of equivalence studies.

To the best of our knowledge, there are no software packages or tools that target specifically at the GSDs of equivalence studies. Though some existing software applications, which are not designed for GSD in equivalence studies, can be tricked into doing sequential tests on equivalence hypotheses, the tricking process itself can be confusing and error-prone. Even if the tricking is successful, other

potential problems still exist. First, many of the procedures are based on the large sample normality assumption. Second, the futility boundaries are calculated from testing $H_0: \theta = 0$. Since $\theta = 0$ represents perfect equivalence, rejecting $\theta = 0$ to claim futility is inappropriately conservative and results in inflated Type II error rate since any other $\theta \in (L, U)$ per the equivalence hypothesis also falls within the range of “equivalence”. Third, the existing GSD software packages, because they are not designed for equivalence studies, can only accommodate symmetric equivalence bounds about 0: $(-\Delta, \Delta)$, which can be restrictive in some real life applications.

In this discussion, we introduce the R package **gset** that fills the software gap in GSDs for studies with equivalence hypotheses. The stagewise equivalence and futility boundaries, either binding or nonbinding, are back-calculated from the equations that are formed based on the exact dual t test statistics. The procedure works given any $\theta \in (L, U)$ rather than just $\theta = 0$. Furthermore, symmetry of the equivalence bounds L and U about 0 is not required. In the rest of the paper, we will provide more details on the following topics: the method and computation employed by **gset**, the functions and outputs from the package **gset**, and some examples on applying the package **gset** in designing GSDs in equivalence studies.

Exact test procedure based on bivariate t statistics

Denote by θ the true difference between two groups of normally distributed variables Y_1 and Y_2 . The estimate of θ is denoted by $\hat{\theta}$; and its standard error is $\sigma_{\hat{\theta}}$, which is estimated by s . If the original data deviates significantly from the normal distribution, an appropriate transformation can be used, and θ would be the difference on the transformed scale. A familiar example is the area under the concentration curve (AUC) and the maximum concentration (Cmax) following a drug administration in bioequivalence studies, which are often modeled by the log-normal distribution. The comparisons between two groups on AUC and Cmax are often performed on the log-scale after the log transformation.

The t statistics for testing the dual hypotheses H_{10} and H_{20} in equivalence studies (in the non-sequential testing) are $T(L) = (\hat{\theta} - L)s^{-1}$ and $T(U) = (\hat{\theta} - U)s^{-1} = \frac{(\hat{\theta}-L)/\sigma_{\hat{\theta}}+(L-U)/\sigma_{\hat{\theta}}}{(s/\sigma_{\hat{\theta}})}$, respectively. $\{T(L), T(U)\}$ jointly follow a bivariate non-central t distribution; and marginally, $T(L)$ follows a central t distribution and $T(U)$ follows a non-central t distribution with non-centrality parameter $(L - U)/\sigma_{\hat{\theta}}$. A decision rule would reject both H_0 simultaneously and should control the overall Type I error rate at a nominal level α . In the sequential test case, the dual test statistics are denoted by $T_k(L) = (\hat{\theta}_k - L)/s_k$ and $T_k(U) = (\hat{\theta}_k - U)/s_k$, where $\hat{\theta}_k$ and s_k are computed with the data accrued up to stage k ($k = 1, \dots, K$; K is the total number of looks). At an interim look k ($k < K$), there are 3 stopping options: a) stop the trial and reject both H_{10} and H_{20} to claim equivalence; b) stop the trial and “reject” either H_{21} or H_{22} to conclude futility; c) move on to the next stage if the test statistics do no cross either the equivalence or the futility boundaries. The rejection region \mathcal{R}_k in stage k (to claim equivalence), acceptance region \mathcal{A}_k (to conclude futility), and the continuation region \mathcal{C}_k are:

$$\mathcal{R}_k = \{T_k(L) > c_k^L\} \cap \{T_k(U) < c_k^U\}, \tag{1}$$

$$\mathcal{A}_k = \{T_k(L) < d_k^L\} \cup \{T_k(U) > d_k^U\}, \tag{2}$$

$$\mathcal{C}_k = (\mathcal{R}_k \cup \mathcal{A}_k)^c, \tag{3}$$

where (c_k^L, c_k^U) are the equivalence boundaries, and (d_k^L, d_k^U) are the futility boundaries. In the last stage K , $\Pr(\mathcal{C}_K)$ is necessarily 0 to allow a final dichotomous decision made between rejection and acceptance of non-equivalence.

The futility boundaries can be binding or non-binding, depending on whether the equivalence boundaries are affected by the action of stopping for futility or not. When the futility boundaries are binding, the decisions to stop for equivalence or futility are “competitive”; that is, the trial will stop at the time point whichever boundaries are crossed first. Otherwise, the overall Type I error rate would be inflated. The equivalence and binding futility boundaries can be determined from the following equation system

$$\Pr(\mathcal{R}_1 | \theta = L) = \Pr(\mathcal{R}_1 | \theta = U) = \alpha(t_1) \tag{4}$$

$$\Pr(\mathcal{C}_1 \cap \dots \cap \mathcal{C}_{k-1} \cap \mathcal{R}_k | \theta = L) = \Pr(\mathcal{C}_1 \cap \dots \cap \mathcal{C}_{k-1} \cap \mathcal{R}_k | \theta = U) = \alpha(t_k) - \alpha(t_{k-1}) \tag{5}$$

$$\Pr(\mathcal{A}_1 | \theta \in (L, U)) = \beta(t_1) \tag{6}$$

$$\Pr(\mathcal{C}_1 \cap \dots \cap \mathcal{C}_{k-1} \cap \mathcal{A}_k | \theta \in (L, U)) = \beta(t_k) - \beta(t_{k-1}), \tag{7}$$

where $k = 2, \dots, K$ in Equations (5) and (7). $\alpha(t_k)$ and $\beta(t_k)$ are the cumulative Type I and II error rates at stage k that can be conveniently specified using the error spending functions (Lan and DeMets, 1983; Hwang et al., 1990). The α - and β -spending functions do not have to be the same.

In many practical cases, the study sponsors would want an option to continue the trial even when the interim data suggests stopping for futility. The futility boundaries calculated under this circumstance are referred to as “non-binding”. A study with non-binding futility boundaries can stop for futility at an interim look when the futility boundaries are crossed, or not. Non-binding futility provides more flexibility to a study without inflating Type I error rate. The equivalence and non-binding futility can be determined from Equations (4), (6), and (7), plus the equation below

$$\Pr(\mathcal{R}_1^c \cap \dots \cap \mathcal{R}_{k-1}^c \cap \mathcal{R}_k \mid \theta = L) = \Pr(\mathcal{R}_1^c \cap \dots \cap \mathcal{R}_{k-1}^c \cap \mathcal{R}_k \mid \theta = U) = \alpha(t_k) - \alpha(t_{k-1}). \quad (8)$$

In terms of the actual computation of the critical values, **gset** employs the Monte Carlo (MC) simulation approach to calculate equivalence and futility boundaries (either non-binding or binding) from the two equation systems given above. Liu and Li (2014) prove that there are some inherent constraints among the critical values including $c_k^U + c_k^L = 0$ in both the binding and non-binding cases, as well as $d_k^L + d_k^U = 0$ in the non-binding case. In the binding case, some constraint has to be imposed on (d_k^L, d_k^U) in order to obtain unique solutions on the boundaries since the the number of unknowns outnumber the number of equations. We adopt the probability symmetry constraint as suggest by Liu and Li (2014). That is, the likelihood that equivalence is rejected due to rejecting H_{10} is the same as the likelihood of rejecting from H_{20} . It turns out that the probability symmetry constraint can be reduced to the simple constraint $d_k^L = -d_k^U$ as in the non-binding case if $(\theta - L)\sigma_\theta^{-1} \gg 1$ and $(U - \theta)\sigma_\theta^{-1} \gg 1$, which is the case in many real-life situations. With the appropriate constraints in place, the boundaries in the non-binding case are calculated in two steps: first solve for (c_k^L, c_k^U) as if there were no futility boundaries, and then compute (d_k^L, d_k^U) conditional on the calculated (c_k^L, c_k^U) from the previous step (Cytel, 2009). In the case of binding futilities, c_k and d_k will need to be determined simultaneously. After the first look, calculation of (c_k^L, c_k^U) will be conditional on the values (d_k^L, d_k^U) computed from the previous step, and vice versa.

In the last step, the equivalence and futility boundaries must meet so that a final dichotomous decision on whether equivalence is achieved can be made. If the originally planned n is not large enough, the two types of boundaries will not meet. One simple approach is to force $|d_k^L| \equiv |d_k^U| = |c_k^L| \equiv |c_k^U|$, with the side effect of inflating the Type II error rate. An alternative, which is better, is to calculate the minimum required n that guarantees that the equivalence and futility boundaries will meet in the last step. In **gset**, this minimum required n is denoted by n_{minimax} , “max” because n_{minimax} would be the maximum used subjects of a study if it makes it into the last stage K . In **gset**, n_{minimax} is searched by a simple bisection approach, with a user-supplied search region.

Implementation in R

The package **gset** contains 8 functions (Table 1). Among the 8 functions, 4 functions can be used to compute the equivalence and futility boundaries (equivonly, nonbinding, binding, nminmax). The futility critical values calculated by nonbinding and binding in the last step are forced to equal to the equivalence critical values by default so that a dichotomous decision on equivalence can be made in the last stage, though users can use argument `force = FALSE` to not force them to be the same. The sample size where the futility and equivalence critical values naturally coincide in the last stage are calculated by function `nminmax`, with the nominal Type I and II error rates maintained. In other words, the futility and equivalence boundaries agree naturally in the last stage with n_{minimax} . Different sample sizes would only affect the futility boundaries, but not the equivalence boundaries. One function (`nf ix`) computes the sample size of an equivalence study in the traditional non-sequential setting. Two functions generate the stagewise boundary plots (`figureE`, `figureEF`); and one function (`oc`) examines the operating characteristics of a given GSD in equivalence studies; including the empirical Type I error rate, empirical power, expected sample size, and the probabilities of stopping at interim looks due to equivalence or futility.

Table 2 lists the arguments to be supplied by users for calculating equivalence and futility boundaries. To calculate the sample size of an equivalence study in the traditional non-sequential setting via `nf ix`, besides `l`, `u`, `theta`, `sigma`, `type1`, `type2` as listed in Table 2, users also need to supply the sampling ratio between the two groups: `r`, and a 2-dimensional vector containing the end-points of the interval from which the sample size will be solved: `nrange`; the default is `c(0, 1000)`. To calculate the sample size of an equivalence study in the GSD setting via `nminmax`, besides `l`, `u`, `theta`, `sigma`, `t.vec`, `type1`, `type2`, `gamma`, `n.sim` as given in Table 2, users also need to specify a logical argument `binding`: whether the futility boundaries are binding, and `n1.lower`, `n1.upper`, `n2.lower`, `n2.upper`, which represent the lower and upper bounds of the interval from which `nminmax` in groups 1 and 2 will be solved using a bisection method. The boundary plots are generated directly from functions `equivonly`, `nonbinding` and `binding` by default (users can suppress the plots by specifying `binding = FALSE`), or they can be generated by functions `figureE` and `figureEF` by taking the boundary outputs

Function	Description
equivonly	computes equivalence boundaries for GSD in equivalence studies that only stop for equivalence
nonbinding	computes equivalence and non-binding futility boundaries for GSD in equivalence studies
binding	computes equivalence and binding futility boundaries for GSD in equivalence studies
nminmax	calculates the sample size of an equivalence study in the sequential setting as well as the equivalence and futility boundaries (either binding or non-binding) under the calculated sample size
nfix	calculates the sample size of an equivalence study in the traditional non-sequential setting
figureE	generates the stagewise equivalence boundary plots
figureEF	generates the stagewise equivalence and futility boundary plots
oc	examines the operating characteristics of a given GSD in equivalence studies

Table 1: Functions in package `gset`.

Argument	Description
l	the lower equivalence bound as given in the equivalence hypothesis
u	the upper equivalence bound as given in the equivalence hypothesis
theta	the true mean difference between 2 groups
sigma	between-subject standard deviation of the response variable y for two independent samples; and within-subject SD of y for paired samples.
n1	size of group 1
n2	size of group 2
t.vec	cumulative interim look time points assuming a constant accrual rate. For example, if a study has equally spaced 4 looks, then $t.vec = 1:4/4$. $t.vec$ can be any vector as long as it is monotonically increasing and the last element is 1
type1	overall Type I error rate
type2	overall Type II error rate
gamma	the gamma parameter in the gamma cumulative error spending function. gamma is a scalar for <code>equivonly</code> , and a 2-dimensional vector for <code>binding</code> , <code>nonbinding</code> , <code>nminmax</code> ; it can be any value; default is -4 , which produces O'Brien-Fleming type error spending function.
crange	a 2-dimensional vector containing the end-points of the interval from the equivalence boundaries will be solved; default is $c(-10, 10)$.
drange	a 2-dimensional vector containing the end-points of the interval from which the futility boundaries will be solved; default is $c(-10, 10)$.
force	whether to force the futility boundaries to equal to the equivalence boundaries in the last look; default is <code>force = TRUE</code> .
plot	whether to generate the boundaries plot. Default <code>plot = TRUE</code> .
ll	a parameter in the boundary plot; the short arm of the $t(L)$ and $t(U)$ axes
ul	a parameter in the boundary plot; the long arm of the $t(L)$ and $t(U)$ axes
n.sim	number of randomly simulated samples in the MC computation of the boundaries; default $n.sim = 10^5$
seed	seed used in the MC computation. If non-specified, the seed is set randomly.

Table 2: Arguments to be supplied by users to calculate equivalence and futility boundaries.

from `equivonly`, `nonbinding`, `binding`, `nminmax` as their input.

As for the error spending function employed by `gset`, the gamma error spending function as introduced by Hwang et al. (1990) is used on both Type I and Type II error. The function, using Type I error rate α as an example, is $f(t) = \alpha(1 - e^{-t\gamma})(1 - e^{-\gamma})^{-1}$ if $\gamma \neq 0$ or αt if $\gamma = 0$. The values of γ can be different for α and β spending. The error spending function is versatile and yield a wide range of spending patterns by varying γ on a continuous scale. For example, $\gamma = -4$ corresponds to more conservative spending at early stages and gradually become liberal toward the end (producing O'Brien-Fleming like boundaries), and $\gamma = 1$ corresponds to more spending at early stages than the later stages (producing Pocock-like boundaries).

Examples

In this section, we illustrate the implementation of package `gset` with 3 examples. In all examples, the base design is the complete crossover design, there are 4 looks of the GSD in each example. The equivalence bounds in the hypothesis are $(L, U) = (-0.2, 0.2)$, the assumed true different $\theta = 0$, the within-subject standard deviation $\sigma = 0.4$, and the overall Type I and Type II error rates are 0.05 and 0.2, respectively. The O'Brien-Fleming type error spending function is used for both the Type I and Type II error rates ($\gamma = -4$), which is the default. Since the critical values and the sample size in `gset` are calculated via the MC simulation, there will be some MC errors in the results. In other words, the results will be slightly different when re-running the commands. To decrease the MC errors, the number of simulations can be increased; the default `n.sim` is 10^5 . Users can fix the random effect using `set.seed` to have their results reproduced in different runs.

```
library(gset)
#### specify the parameters
L <- -0.2
U <- 0.2
theta <- 0
sigma <- 0.4
alpha <- 0.05
beta <- 0.2
K <- 4
r <- 1
#### sample size in the non-sequential setting
n.fix <- nfix(r, L, U, theta, sigma, alpha, beta)
```

The output is given below. The sample size is 69 for a non-sequential crossover design.

```
$n1
[1] 69
$n2
[1] 69
```

Example 1: If a study considers only stopping for equivalence, then the following command computes the equivalence boundaries and generates the boundary plots.

```
bound1 <- equivonly(L, U, sigma, n.fix$n1, n.fix$n2, 1:K/K, alpha, beta)
#### the boundary plot can be regenerated by using figureE(bound1, K)
```

The output is given below. It contains the cumulative Type I error spending and the equivalence critical values at each look. If $T(L) > 1.858$ and $T(U) < -1.858$ at the first interim look of a GSD, the study stops for equivalence; otherwise, the study continues. The usage of the critical values at other looks are similar. The boundary plots are given in Figure 1.

```
$typeI
[1] 0.001602930 0.005960146 0.017804287 0.050000000
$equivL
[1] 1.857541 2.151003 2.212045 1.739736
$equivU
[1] -1.857541 -2.151003 -2.212045 -1.739736
```

The operating characteristics of the GSD with the calculated equivalence boundaries can be investigated by applying commands `oc(L,U,theta = L,sigma,K,69,69,bound1,futility = FALSE)` (under H_0) and `oc(L,U,theta = 0,sigma,K,69,69,bound1,futility = FALSE)` (under H_1). The outputs are not provided due to space limitation.

Example 2: If a study considers stopping for equivalence and futility but would like to have the flexibility to continue even if the futility boundaries are crossed at an interim look, then the following commands can be used to get the critical values for a GSD with non-binding futility. By default, the futility boundaries in the last step are forced to equal to the equivalence boundaries; users can use argument `force = FALSE` to remove the constraint.

```
bound2 <- nonbinding(L, U, theta, sigma, n.fix$n1, n.fix$n2, 1:K/K, alpha, beta)
### the boundary plot can be regenerated by using figureEF(bound2, K)
```

The output is given below. It contains the cumulative error spending and the equivalence and non-binding futility boundaries at each look. Note that the equivalence critical values with non-binding

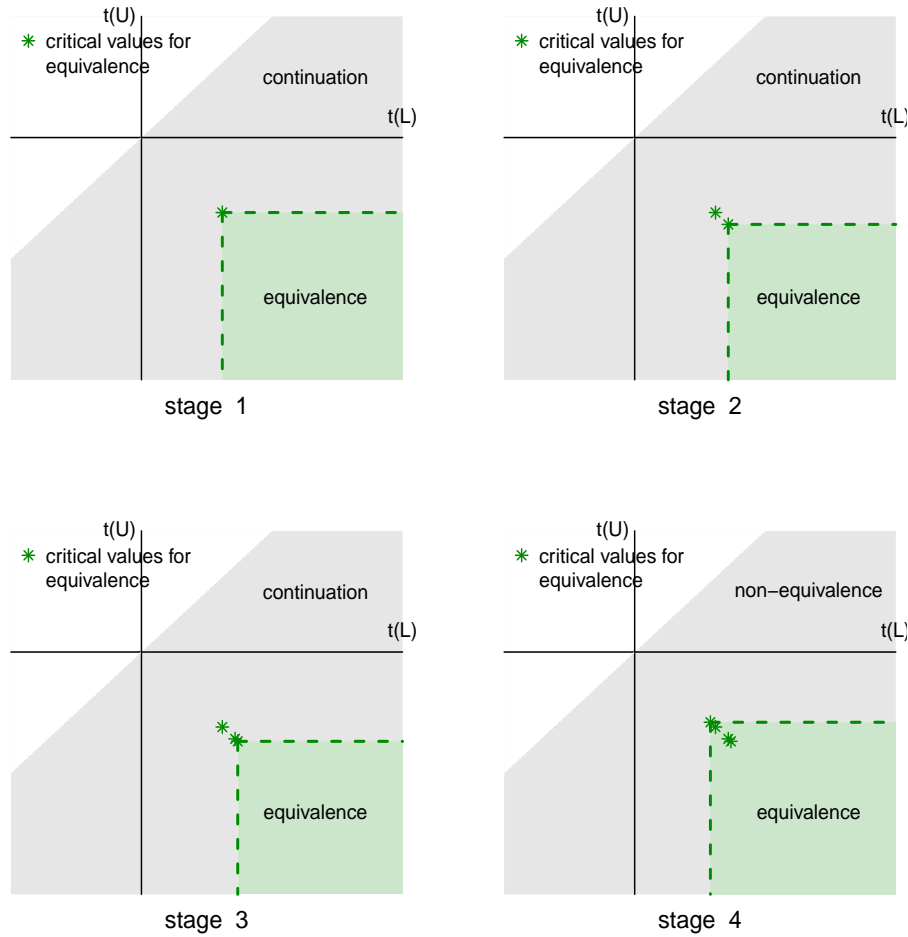


Figure 1: Stagewise boundary plot in Example 1.

futility in theory should be the same as the equivalence critical values in a study with equivalence boundaries only. Comparing the results from equivalently above, we can see that the critical values are close but not exactly the same due to the MC errors. If the same random seed had been used, the results would have been the same.

```

$typeI
[1] 0.001602930 0.005960146 0.017804287 0.050000000
$typeII
[1] 0.006411721 0.023840584 0.071217148 0.200000000
$equivL
[1] 1.829343 2.160757 2.197812 1.721998
$equivU
[1] -1.829343 -2.160757 -2.197812 -1.721998
$futilL
[1] -1.2607491 -0.2314749 0.6712118 1.7219983
$futilU
[1] 1.2607491 0.2314749 -0.6712118 -1.7219983
    
```

The boundary plots are given in Figure 2. Since $T(L) > T(U)$, the region above the identity line $T(U) = T(L)$ is impossible. In this particular example, the futility critical values at the 1st and 2nd interim looks appear above the identity line (in the 2nd quadrant, specifically), implying that the trial cannot stop for futility in the first two looks. The operating characteristics of the GSD with the calculated equivalence and futility boundaries can be investigated by applying commands `oc(L,U,theta = L,sigma,K,69,69,bound2,futility = TRUE)` (under H_0) and `oc(L,U,theta = 0,sigma,K,69,69,bound2,futility = TRUE)` (under H_1). The outputs are given in Table 3. The empirical Type I error (0.048) is maintained at the nominal level (5%). At $n = 69$, the study is a little underpowered (empirical power = 78.7%, below the desired level 80%). The expected sample size is smaller under H_1 compared to that under H_0 but close, and both are smaller than the fixed sample size $n = 69$. The probability of stopping due to equivalence at look 3 when H_1 is true is large (25.4%),

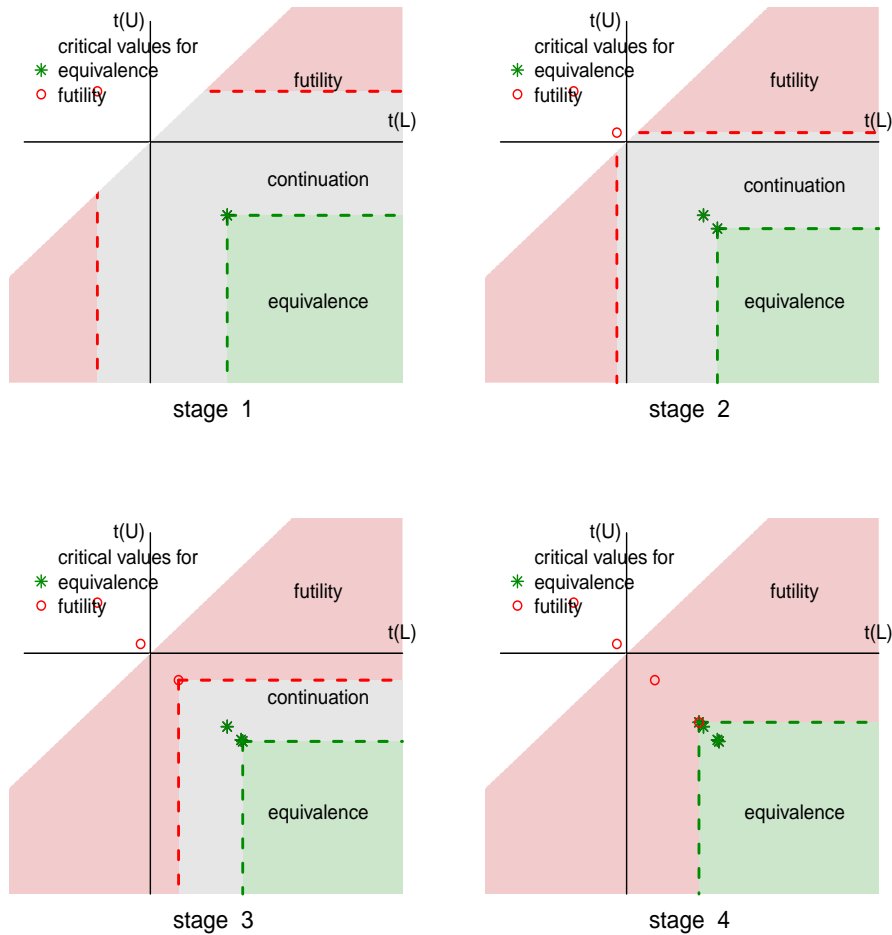


Figure 2: Stagewise boundary plot in Example 3.

under H_0	under H_1
\$reject.rate	\$reject.rate
[1] 0.0479	[1] 0.787275
\$accept.rate	\$accept.rate
[1] 0.9521	[1] 0.223075
\$En1	\$En1
[1] 66.9	[1] 62.3
\$En2	\$En2
[1] 66.9	[1] 62.3
\$prob.stop	\$prob.stop
[1] 0.008025 0.022650 0.055450 0.913875	[1] 0.011450 0.052500 0.299125 0.636925
\$prob.stopE	\$prob.stopE
[1] 0.002075 0.004375 0.010900 0.030550	[1] 0.005325 0.034550 0.253925 0.493475
\$prob.stopF	\$prob.stopF
[1] 0.005950 0.018275 0.044550 0.883325	[1] 0.006125 0.017950 0.045200 0.153800

Table 3: Output from command oc on the operating characteristics of the GSD under H_0 and H_1 in Example 2 when $n = 69$.

but not at other looks. This is partly due to the conservative spending of the Type I error rate in early stages of the O'Brien-Fleming type error spending, making it harder to reject early, especially with the slight under-power.

To calculate the sample size that yields the desirable power (80%), the following command can be used. The equivalence boundaries should remain the same (except than some MC numerical errors) as those under $n = 69$, but the futility boundaries would alter.

```
bound3 <- nminmax(L, U, theta, sigma, 69, 69, 1:K/K, alpha, beta)
```


The outputs are given below.

```

$n1minmax
[1] 75
$n2minmax
[1] 75
$typeI
[1] 0.001602930 0.005960146 0.017804287 0.050000000
$typeII
[1] 0.006411721 0.023840584 0.071217148 0.200000000
$equivL
[1] 1.852941 2.193359 2.208099 1.729116
$equivU
[1] -1.852941 -2.193359 -2.208099 -1.729116
$futilL
[1] -1.2006119 -0.1403367 0.8067496 1.7291157
$futilU
[1] 1.2006119 0.1403367 -0.8067496 -1.7291157

```

The new sample size is 75. The boundary plot, which can be obtained using `figureEF(bound3,K)`, is similar to Figure 2, and not provided due to space limitation. The operating characteristics of the GSD with $n = 75$ can be also investigated by applying the following commands `oc(L,U,theta = L,sigma,K,75,75,bound3,futility = FALSE)` (under H_0) and `oc(L,U,theta = 0,sigma,K,75,75,bound3,futility = FALSE)` (under H_1). Due to space limitation, the output is not shown. The empirical power with $n = 75$ now increases to 82.7%. The probabilities of stopping due to equivalence in early stages also increase under H_1 . The empirical Type I error rate remains controlled at 0.05 under H_0 .

Example 3: If a study plans to stop the study whenever the equivalence or the flexibility boundaries are crossed at an interim look, then the following command can be used. By default, the futility boundaries in the last step are forced to equal to the equivalence boundaries; users can use argument `force = FALSE` to remove the constraint.

```

bound4 <- binding(L, U, theta, sigma, n.fix$n1, n.fix$n2, 1:K/K, alpha, beta)
### the boundary plot can be regenerated by using figureEF(bound4, K)

```

The output is given below. The equivalence critical values (`binding futility`) are different from those from studies with nonbinding futilities (Example 2). The boundary plots are given in Figure 3. The futility critical values at the first and second interim looks appear above the identity line in this example, and the trial cannot stop for futility in the first two looks.

```

$typeI
[1] 0.001602930 0.005960146 0.017804287 0.050000000
$typeII
[1] 0.006411721 0.023840584 0.071217148 0.200000000
$equivL
[1] 1.806721 2.155557 2.220632 1.730838
$equivU
[1] -1.806721 -2.155557 -2.220632 -1.730838
$futilL
[1] -1.2525572 -0.2457909 0.6858693 1.7308381
$futilU
[1] 1.2525572 0.2457909 -0.6858693 -1.7308381

```

The operating characteristics of the GSD with the equivalence and binding futility boundaries can be investigated by applying commands `oc(L,U,theta = L,sigma,K,69,69,bound4,futility = TRUE,binding = TRUE)` (under H_0) and `oc(L,U,theta = 0,sigma,K,69,69,bound4,futility = TRUE,binding = TRUE)` (under H_1). The output is given in Table 4. The empirical Type I error (0.049) is maintained at the nominal level (5%). At $n = 69$, the GSD study is a bit underpowered (empirical power = 76.8%, below the nominal level 80%). It is interesting to note that the expected sample size is much smaller if H_0 is true compared to that under H_1 , which is exactly the opposite than the non-binding case though the difference is not as dramatic. This is because the study will have to stop if the futility boundaries are crossed under H_0 .

To calculate the sample size that yields the desirable power (80%) in Example 3, the following command can be used. The equivalence boundaries should remain the same (except for some MC numerical errors) as those under $n = 60$, but the futility boundaries would alter.

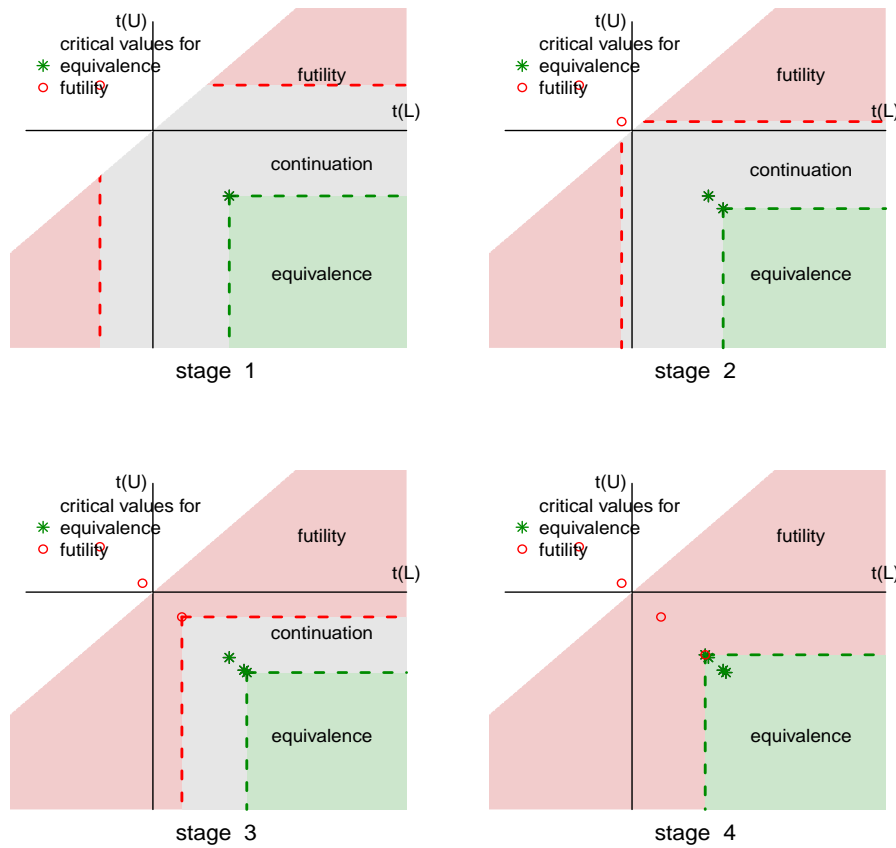


Figure 3: Stagewise boundary plot in Example 3.

under H_0	under H_1
\$reject.rate	\$reject.rate
[1] 0.0491	[1] 0.76775
\$accept.rate	\$accept.rate
[1] 0.9509	[1] 0.23225
\$En1	\$En1
[1] 46.7	[1] 61.9
\$En2	\$En2
[1] 46.7	[1] 61.9
\$prob.stop	\$prob.stop
[1] 0.114075 0.306725 0.354850 0.224350	[1] 0.011675 0.052475 0.275500 0.660350
\$prob.stopE	\$prob.stopE
[1] 0.002075 0.004200 0.011050 0.031775	[1] 0.005325 0.036075 0.227675 0.498675
\$prob.stopF	\$prob.stopF
[1] 0.112000 0.302525 0.343800 0.192575	[1] 0.006350 0.016400 0.047825 0.161675

Table 4: Output from command oc on the operating characteristics of the GSD under H_0 and H_1 in Example 3 when $n = 69$.

```
bound5 <- nminmax(L, U, theta, sigma, 69, 69, 1:K/K, alpha, beta, binding = TRUE)
```

The outputs are given below. The new sample size is 73. The boundary plot, which can be obtained using figureEF(bound5,K) is not provided due to space limitation. The operating characteristics of the GSD with $n = 73$ can be investigated by applying commands oc(L,U,theta = L,sigma,K,73,73,bound5,futility = FALSE,binding = TRUE) (under H_0) and oc(L,U,theta = 0,sigma,K,73,73,bound5,futility = FALSE,binding = TRUE) (under H_1). Due to space limitation, the output is not shown. The empirical power with $n = 75$ increases to 80.3%.

```
$n1minmax
[1] 73
$n2minmax
[1] 73
```

```

$typeI
[1] 0.001602930 0.005960146 0.017804287 0.050000000
$typeII
[1] 0.006411721 0.023840584 0.071217148 0.200000000
$equivL
[1] 1.881127 2.187860 2.217254 1.716097
$equivU
[1] -1.881127 -2.187860 -2.217254 -1.716097
$futilL
[1] -1.2035253 -0.1731546 0.7543735 1.7160975
$futilU
[1] 1.2035253 0.1731546 -0.7543735 -1.7160975

```

Conclusion

We have introduced the R package **gset** that computes the stagewise critical values and sample size for testing equivalence hypothesis in GSDs. We outlined the underlying theory and computation approach that **gset** is based on, and illustrated the usage of the package with several GSD examples. **gset** can compute the critical values for GSDs that stop only for equivalence, or stop for equivalence and futility – either binding or non-binding. It also produces 2-dimensional boundary plots which give a direct visualization of the stagewise stopping boundaries. The operating characteristics of the a proposed GSD can be examined in **gset** via the computation of empirical Type I error rates, empirical power, stopping probabilities at the interim looks, and expected sample sizes.

gset is the first package in R that targets specifically at the GSD with equivalence hypothesis. Furthermore, it is based on the exact bivariate t test statistics, making it a fitting choice for GSDs with small (stagewise) sample size cases and small stagewise Type I error rates. **gset** is based on the traditional GSD framework and it does not accommodate sample size re-estimation during a study when interim data becomes available, a feature of “flexible” GSDs. **gset** assumes the analyzed variables follow a normal distribution. If there is severe deviation from normality, transformation can be applied before applying the package.

Bibliography

- K. Anderson. *gsDesign: Group Sequential Design*, 2014. URL <http://CRAN.R-project.org/package=gsDesign>. R package version 2.9-3. [p174]
- Cytel. *EaST for Windows: A Software Package for the Design and Analysis of Group Sequential Clinical Trials*. Cytel Software Corporation, Cambridge, Massachusetts, 2009. [p176]
- S. Durrleman and R. Simon. Planning and monitoring of equivalence trials. *Biometrics*, 46(2):329–336, 1990. [p174]
- L. A. Gould. Group sequential extensions of a standard bioequivalence testing procedure. *Journal of Pharmacokinetics and Biopharmaceutics*, 23(1):57–86, 1995. [p174]
- N. Hack, W. Brannath, and M. Brueckner. *AGSDest: Estimation in Adaptive Group Sequential Trials*, 2013. URL <http://CRAN.R-project.org/package=AGSDest>. R package version 2.1. [p174]
- F. E. Harrell Jr. *Hmisc: Harrell Miscellaneous*, 2014. URL <http://CRAN.R-project.org/package=Hmisc>. R package version 3.14-6. [p174]
- Y. Huang and S. G. Self. A comparison of testing parameters and the implementation of a group sequential design for equivalence studies using paired-sample analysis. *Statistics in Pharmaceutical Research*, 2(3):336–347, 2010. [p174]
- I. K. Hwang, W. J. Shih, and J. S. DeCani. Group sequential designs using a family of type I error probability spending functions. *Statistics in Medicine*, 9(12):1439–1445, 1990. [p175, 177]
- J. Hwang. Numerical solutions for a sequential approach to bioequivalence. *Statistica Sinica*, 6(3): 663–673, 1996. [p174]
- G. Izmirlan. *PwrGSD: Power in a Group Sequential Design*, 2014. URL <http://CRAN.R-project.org/package=PwrGSD>. R package version 2.000. [p174]

- C. Jennison and B. Turnbull. Interim analyses: The repeated confidence interval approach (with discussion). *Journal of the Royal Statistical Society, Series B*, 51(3):305–361, 1989. [p174]
- C. Jennison and B. Turnbull. Group sequential tests for bivariate response: Interim analyses of clinical trials with both efficacy and safety endpoints. *Biometrics*, 49(3):741–752, 1993. [p174]
- K. Lan and D. DeMets. Discrete sequential boundaries for clinical trials. *Biometrika*, 70(3):659–663, 1983. [p175]
- F. Liu and Q. Li. Exact sequential test of equivalence hypothesis based on bivariate non-central t -statistics. *Computational Statistics and Data Analysis*, 77:14–24, 2014. [p174, 176]
- T. H. Montague, D. Potvin, C. E. DiLiberti, W. W. Hauck, A. F. Parr, and D. J. Schuirmann. Additional results for ‘Sequential design approaches for bioequivalence studies with crossover designs’. *Pharmaceutical Statistics*, 11(1):8–13, 2012. [p174]
- R. Pahl. *GroupSeq: A GUI-Based Program to Compute Probabilities Regarding Group Sequential Designs*, 2014. URL <http://CRAN.R-project.org/package=GroupSeq>. R package version 1.3.2. [p174]
- D. Potvin, C. E. DiLiberti, W. W. Hauck, A. F. Parr, D. J. Schuirmann, and R. A. Smith. Sequential design approaches for bioequivalence studies with crossover designs. *Pharmaceutical Statistics*, 7(4): 245–262, 2008. [p174]
- V. E. Seshan. *clinfun: Clinical Trial Design and Data Analysis Functions*, 2014. URL <http://CRAN.R-project.org/package=clinfun>. R package version 1.0.6. [p174]
- J. Whitehead. Sequential designs for equivalence studies. *Statistics in Medicine*, 15(24):2703–2715, 1996. [p174]

Fang Liu
Department of Applied and Computational Mathematics and Statistics
The University of Notre Dame
Notre Dame, IN 46530
USA
fang.liu.131@nd.edu

Conference Report: R in Insurance 2014

by Markus Gesmann, Andreas Tsanakas

Conference summary

The 2nd R in Insurance conference took place at Cass Business School London on 14 July 2014. This one-day conference focused once more on the wide range of applications of R in insurance, actuarial science and beyond. The conference programme covered topics including reserving, pricing, loss modelling, the use of R in a production environment and much more.

The audience of the conference included both practitioners (70%) and academics (30%) who are active or interested in the applications of R in Insurance. It was a truly international event with speakers and delegates from many different countries, including USA, Canada, Belgium, Netherlands, Switzerland, Germany, Ireland, Argentina, France, Spain and of course the UK. The coffee breaks and conference dinner offered great networking opportunities.

In the first plenary session, [Montserrat Guillen](#) (Riskcenter, University of Barcelona) and [Leo Guelman](#) (Royal Bank of Canada, RBC Insurance) spoke about the rise of uplift models. These predictive models are used for improved targeting of policyholders by marketing campaigns, through the use of experimental data. The presenters illustrated the use of their [uplift](#) package (available on CRAN), which they have developed for such applications.

Thereafter, the programme consisted of a combination of contributed presentations and lightning talks, as well as a panel discussion on R at the interface of practitioner / academic interaction. The panel, drawn from academia and practice, discussed the efforts made in bridging through the use of R cultural and communication divides, as well as the challenges of developing collaborative business models that respond to market needs and the incentives of academic researchers.

In the closing plenary, [Arthur Charpentier](#) (Professor of Actuarial Science at UQAM, Canada) gave a non-Bayesian's account of Bayesian modelling in R. Bayesian philosophy has a long history in actuarial science, even if it is sometimes hidden. The presenter demonstrated how some standard actuarial methods can be expressed in a Bayesian modelling framework using R.

All conference presentations are available on request, contact: info@rininsurance.com.

Scientific committee and sponsors

The members of the scientific committee were: [Katrien Antonio](#) (University of Amsterdam and KU Leuven), [Christophe Dutang](#) (Universite du Maine, France), [Jens Nielsen](#) (Cass Business School), [Andreas Tsanakas](#) (Cass Business School) and [Markus Gesmann](#) (ChainLadder project)

Finally, we are grateful to our sponsors [Mango Solutions](#), [CYBAEA](#), [PwC](#) and [RStudio](#). This conference would not have been possible without their generous support.

R in Insurance 2015

We are delighted to announce next year's event already. Following two years in London at Cass Business School, the conference will travel across the Channel to Amsterdam, 29 June 2015. Further details will be published on www.rininsurance.com.

*Markus Gesmann
ChainLadder project
London*

UK

markus.gesmann@gmail.com

Andreas Tsanakas

Cass Business School

London

UK

a.tsanakas.1@city.ac.uk

Conference Report: Polish Academic R User Meeting

by Maciej Beręsewicz, Alicja Szabelska, Joanna Zyprych-Walczak and Łukasz Wawrowski

Conference summary

The first national conference “Polish Academic R User Meeting” (PAZUR) was held at the Poznan University of Economics from October 15–17, 2014. The organizers of the conference were the Department of Statistics at the Poznan University of Economics (PUE), the Department of Mathematical and Statistical Methods at the Poznan University of Life Sciences (PULS) and SKN Estymator, the students scientific association that resides at the Department of Statistics at the Poznan University of Economics. The honorary patronage of the conference took Professor Emil Panek, Dean of the Faculty of Informatics and Electronic Economy PUE and Professor Wiesław Koziara, Dean of the Faculty of Agronomy and Bioengineering PULS. In addition, patrons were the Branch in Poznan of the Polish Statistical Association, the Polish Biometric Society and the General Director of the Statistical Office in Poznan. The sponsors of the conference were: [Revolution Analytics Company](#) and [Analyx](#) as well as foundation [SmarterPoland](#).

The main goal of the conference was to present the possibility of using the statistical software environment R in various fields of science and business, exchange experiences and integrate the R users in Poland. The first day of the event was devoted to workshops focusing on spatial data analysis, spatial data mining, processing large data sets with the [data.table](#) package (Dowle et al., 2014), survey data analysis with the [survey](#) package (Lumley, 2004, 2014), data visualization, as well as the introduction to the analysis of data in the statistical software environment R. The workshop was attended by nearly 130 participants. During the second and third day of the conference 29 presentations concerning the variety of applications of R in the fields of mathematics, biological sciences, economics and information technology as well as business applications. The participation in the conference was free of charge. There were two invited talks during the conference presented by: dr hab. Katarzyna Kopczewska (Faculty of Economic Sciences, University of Warsaw) entitled *Spatial modelling of economic phenomenon in R* (in Polish: *Przestrzenne modelowanie zjawisk ekonomicznych w R*) and dr hab. Przemysław Biecek (Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw) entitled *Project Beta and Bit, ULAM diagrams, data visualisation in R – my analytical atelier* (in Polish: *Projekt Beta i Bit, diagramy ULAM, wizualizacja z R, czyli o moim analitycznym atelier*).

All the presentations from the conference are available at the webpage www.estymator.ue.poznan.pl/PAZUR and [here](#).

The organizing committee

The members of the organizing committee were: Maciej Beręsewicz (chairperson, Poznan University of Economics, Poznan Statistical Office), Joanna Zyprych-Walczack (Poznan University of Life Sciences), Alicja Szabelska (Poznan University of Life Sciences) and Łukasz Wawrowski (Poznan University of Economics, Poznan Statistical Office).

The presentations

Beside the two invited talks the conference participants presented the following 27 topics (in alphabetical order):

- Adolfo Alvarez (Analyx): Mastering data frames with dplyr

- Zbigniew Bartkowiak (Poznan Archaeological Museum): The impact of trimming and mixing ore of Wladyslaw Jagiello's coins on the distribution of their weight
- Maciej Beręsewicz (Poznan University of Economics): Small area statistics in R
- Paweł Budzianowski (Adam Mickiewicz University): A copula-based multivariate analysis with R
- Damian Chmura (Academy of Humanities and Technology in Bielsko-Biala): The use of the R package in teaching courses of biocoeno in the fields of study: Environmental protection and environmental engineering
- Marcin K. Dyderski, Anna K. Gdula (Botanical Section students scientific foresters association in Poznan University of Life Sciences): Analysis of vegetation using vegan and eHOF packages
- Marek Gałęwski (Systems Research Institute, Polish Academy of Sciences): Processing string using stringi package and ICU libraries
- Jarosław Jasiewicz (Adam Mickiewicz University): GIS-R – Integrating geographic information systems and R
- Paweł Kliber (Poznan University of Economics): R in the financial analysis
- Paweł Kleka (Adam Mickiewicz University): From the data to publication. Automatic formatting of the results using knitr package
- Krzysztof Krawiec (Poznan University of Technology): Genetic programming
- Marcin Kosiński (Warsaw University of Technology): Archiving artifacts with the Archivist package
- Jakub Nowosad (Adam Mickiewicz University): Visualization of spatial data in R – from static to interactive map
- Michał Okoniewski (ETH Zurich, Scientific IT Services): R with external data sources in genomics – Nonsense case study
- Adrian Olszewski (KCR): GNU R in clinical trials – biostatistician workspace
- Natalia Reszka (PBS): Integrating R and NetLogo agent systems simulations
- Kamil Rotter (SKN Estimator, Poznan University of Economics): Internet as a source of data on the example of the Allegro service
- Artur Schwałko (QuantUp): If / how R can replace Excel in the company?
- Agnieszka Strzałka (University of Wrocław): Using ggplot2 to visualize the growth of *Streptomyces coelicolor*
- Alicja Szabelska, Joanna Zyprych-Walczak (Poznan University of Life Sciences): Using R in the analysis of RNA-seq data
- Monika Szczerba, Marek Wiewiórka (Warsaw University of Technology): R + big (bio)data warehouses
- Łukasz Wawrowski (Poznan University of Economics): Data analysis from social networking sites
- Marek Wiewiórka (Warsaw University of Technology): Scala(-ble) R – R integration and the Scala language
- Marek Wiewiórka (Warsaw University of Technology), Alicja Szabelska (Poznan University of Life Sciences), Michał Okoniewski (ETH Zurich, Scientific IT Services): Disease gene signature – the use of parallel R and machine learning with MLInterfaces
- Adam Zagański (Wrocław University of Technology/QuantUp): Analysis and forecasting of time series in R
- Zygmunt Zawadzki, Daniel Kosiorowski (Cracow University of Economics): Robust data mining using depthproc package

Bibliography

- M. Dowle, T. Short, S. Lianoglou, A. Srinivasan, and with contributions from R. Saporta and E. Antonyan. *data.table: Extension of data.frame*, 2014. URL <http://CRAN.R-project.org/package=data.table>. R package version 1.9.4. [p187]
- T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. URL <http://www.jstatsoft.org/v09/i01>. [p187]
- T. Lumley. *survey: Analysis of complex survey samples*, 2014. URL <http://CRAN.R-project.org/package=survey>. R package version 3.30. [p187]

Maciej Beręsewicz Department of Statistics
Poznan University of Economics
Poland
maciej.beresewicz@ue.poznan.pl

Alicja Szabelska
Department of Mathematical and Statistical Methods
Poznan University of Life Sciences
Poland
alicja.szabelska@up.poznan.pl

Joanna Zyprych-Walczak
Department of Mathematical and Statistical Methods
Poznan University of Life Sciences
Poland
joanna.zyprych@ue.poznan.pl

Łukasz Wawrowski Department of Statistics
Poznan University of Economics
Poland
lukasz.wawrowski@ue.poznan.pl

R Foundation News

by *Martin Mächler and Kurt Hornik*

New R Foundation board

Subsequent to meetings of the R Foundation in the summer of 2014, and after a careful nomination and voting process, the election of a new board of the R Foundation was finalized on December 1. The new board consists of

Presidents: Duncan Murdoch, Martyn Plummer

Secretary General: Martin Mächler

Treasurer: Kurt Hornik

Members at Large: John Chambers, Brian Ripley, Dirk Eddelbuettel

Peter Dalgaard and Roger Bivand will serve as auditors.

Our many thanks go to the outgoing board members, who have led the R Foundation during the years of phenomenal growth and success since its creation in April 2003: These are Robert Gentleman and Ross Ihaka, our past presidents, and Friedrich Leisch, the former Secretary General.

New ordinary members

The following new ordinary members were elected, as announced on September 15.¹

Dirk Eddelbuettel, USA

Torsten Hothorn, Switzerland

Michael Lawrence, USA

Martin Morgan, USA

Marc Schwartz, USA

Hadley Wickham, USA

Achim Zeileis, Austria

This brings the total number of ordinary members to 29. As always, the complete membership list, along with other details, is available at <http://www.r-project.org/foundation/>.

Donations and new supporting members

Donations

Shannon Callan, USA

Rees Morrison, USA

Carlos Ortega, Spain

New supporting members

Jose M. Benitez, Spain

Daniel Emaasit, USA

Jay Emerson, USA

Bastiaan Quast, Switzerland

¹<https://stat.ethz.ch/pipermail/r-announce/2014/000577.html>

Martin Mächler
ETH Zurich, Switzerland
Martin.Maechler@R-project.org

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Changes on CRAN

2014-07-01 to 2014-12-31

by Kurt Hornik and Achim Zeileis

New packages in CRAN task views

Bayesian BayesTree.

Cluster fclust, funFEM, funHDDC, pgmm, tclust.

Distributions FatTailsR, RTDE, STAR, predfinitepop, statmod.

Econometrics LinRegInteractive, MSBVAR, nonnest2, phtt.

Environmetrics siplab.

Finance GCPM, MSBVAR, OptionPricing, financial, fractal, riskSimul.

HighPerformanceComputing GUIProfiler, PGICA, aprof.

MachineLearning BayesTree, LogicForest, hdi, mlr, randomForestSRC, stabs, vcrpart.

MetaAnalysis MAVIS, ecoreg, ipdmeta, metaplus.

NumericalMathematics RootsExtremaInflections, Rserve, SimplicialCubature, fastGHQuad, optR.

OfficialStatistics CoImp, RecordLinkage, rworldmap, tmap, vardpoor.

Optimization RCEIM, blowtorch, globalOptTests, irace, isotone, lbfgs.

Phylogenetics BAMMtools, BoSSA, DiscML, HyPhy, MPSEM, OutbreakTools, PBD, PCPS, PHYLOGR, RADami, RNeXML, Reol, Rphylip, adhoc, betapart, dendextend, expands, expoTree, jaatha, kdetrees, mvMORPH, outbreaker, pastis, pegas, phyloTop, phyloland, rdryad, rphast, strap, surface, taxize.

Psychometrics IRTShiny, PP, WrightMap, mirtCAT, pairwise.

ReproducibleResearch NMOF.

Robust TEEReg, WRS2, robeth, robustDA, robustgam, robustloggamma, robustreg, ror, rorutadis.

Spatial PRemiuM.

SpatioTemporal BayesianAnimalTracker, TrackReconstruction, fishmove, mkde, wildlifeDI.

Survival DStree, ICsurv, IDPSurvival, MIICD, MST, MicSim, PHeval, PRemiuM, aftgee, bshazard, bujar, coxinterval, gamboostMSM, imputeYn, invGauss, lsmeans, multipleNCC, paf, penMSM, spBayesSurv, survAccuracyMeasures, survivalMPL, vitality.

TimeSeries GMDH, gets, orderedLasso, yuima.

WebTechnologies BerlinData, EIAdata, IBrokers, ONETr, RCryptsy, RDataCanvas, RGA, RGoogleAnalytics, RJSMDX, RPublica, RPushbullet, RSelenium, RStars, SocialMediaMineR, TR8, Taxonstand, W3CMarkupValidator, WikipediR, aRxiv, boilerpipeR, colourlovers, curl, d3Network, dataRetrieval, downloader, ecoretriever, enigma, federalregister, ggvis, gmailr, indicoio, jSonarR, leafletR, magrittr, mailR, marmap, meteoForecast, paleobioDB, polidata, pollstR, pushoverr, pvsR, pxweb, rClinicalCodes, rFDSN, rNOMADS, rWBclimate, rYoutheria, rainfreq, rbitcoin-chartsapi, rclinicaltrials, recalls, redcapAPI, reutils, rnb, rnrf, ropensecretsapi, rsdmx, rsunlight, shopifyr, slackr, sorvi, translateR, tumblr, ustyc, webutils, whisker, wikipediatrend.

New contributed packages

- ADDT** Analysis of Accelerated Destructive Degradation Test Data. Authors: Yili Hong, Yimeng Xie, and Caleb King.
- ARTool** Aligned Rank Transform. Authors: Matthew Kay [aut, cre], Jacob O. Wobbrock [aut].
- ASMap** Linkage map construction using the MSTmap algorithm. Authors: Julian Taylor, David Butler.
- ASPBay** Bayesian Inference on Causal Genetic Variants using Affected Sib-Pairs Data. Author: Claire Dandine-Roulland.
- BANFF** Bayesian Network Feature Finder. Authors: Zhou Lan, Yize Zhao, Jian Kang, Tianwei Yu.
- BAT** Biodiversity Assessment Tools. Authors: Pedro Cardoso, Francois Rigal, Jose Carlos Carvalho.
- BBEST** Bayesian Estimation of Incoherent Neutron Scattering Backgrounds. Authors: Anton Gagin and Igor Levin with contributions from Charles R. Hogg III.
- BEANSP** Bayesian Estimate of Age-specific Nest Survival Probabilities. Authors: Chong He, Yiqun Yang, Jing Cao.
- BHMSMAfMRI** Bayesian hierarchical multi-subject multiscale analysis of functional MRI data. Authors: Nilotpal Sanyal [aut, cre], Marco A.R. Ferreira [aut].
- BIOM.utils** Utilities for the BIOM (Biological Observation Matrix) Format. Author: Daniel T. Braithwaite [aut, cre].
- BNDDataGenerator** Data Generator based on Bayesian Network Model. Author: Jae-seong Yoo.
- BNSP** Bayesian Non- and Semi-parametric Model Fitting. Author: Georgios Papageorgiou.
- BOIN** Bayesian Optimal Interval Design for Phase I Clinical Trials. Authors: Ying Yuan and Suyu Liu.
- BSGW** Bayesian Survival Model using Generalized Weibull Regression. Authors: Alireza S. Mahani, Mansour T.A. Sharabiani.
- BayClone2** Bayesian Feature Allocation Model for Tumor Heterogeneity. Authors: Juhee Lee, Peter Mueller, Subhjit Sengupta, Kamalakar Gulukota, Yuan Ji.
- BayesMixSurv** Bayesian Mixture Survival Models using Additive Mixture-of-Weibull Hazards, with Lasso Shrinkage and Stratification. Authors: Alireza S. Mahani, Mansour T.A. Sharabiani.
- BayesianAnimalTracker** Bayesian Melding of GPS and DR Path for Animal Tracking. Authors: Yang (Seagle) Liu [aut, cre], Brian C. Battaile [ctb]. In view: *SpatioTemporal*.
- BinNonNor** Data Generation with Binary and Continuous Non-normal Components. Authors: Gul Inan, Hakan Demirtas.
- BlandAltmanLeh** Plots (slightly extended) Bland-Altman plots. Author: Bernhard Lehnert.
- CARBAYESST** Poisson Log-linear Models with Spatio-temporal Random Effects. Authors: Duncan Lee and Alastair Rushworth.
- CEC** Cross-Entropy Clustering. Authors: Konrad Kamieniecki [aut, cre], Przemyslaw Spurek [ctb].

- CINOEDV** Co-Information based N-Order Epistasis Detector and Visualizer. Author: Junliang Shang.
- CLME** Constrained Inference for Linear Mixed Effects Models. Author: Casey M. Jelsema.
- CNOGpro** Copy Numbers of Genes in prokaryotes. Authors: Ola Brynildsrud, Lars-Gustav Snipen.
- COPASutils** Tools for processing COPAS large-particle flow cytometer data. Authors: Tyler Shimko, Erik Andersen.
- CP** Conditional Power Calculations. Author: Andreas Kuehnappel.
- CVTuningCov** Regularized Estimators of Covariance Matrices with CV Tuning. Author: Binhuan Wang.
- CarletonStats** Functions For Statistics Classes At Carleton College. Author: Laura Chihara.
- CateSelection** Categorical Variable Selection Methods. Authors: Yi Xu and Jixiang Wu.
- CerioliOutlierDetection** Outlier detection using the iterated RMCD method of Cerioli (2010). Authors: Christopher G. Green [aut, cre], R. Doug Martin [ths].
- ClimClass** Climate Classification According To Several Indices. Authors: Emanuele Eccel, Emanuele Cordano, Giambattista Toller.
- CombinePValue** Combine a Vector of Correlated p -values. Author: Hongying Dai.
- CompGLM** Conway-Maxwell-Poisson GLM and distribution functions. Author: Jeffrey Pollock.
- Compind** Composite indicators functions. Authors: Francesco Vidoli, Elisa Fusco.
- CosmoPhotoz** Photometric redshift estimation using Generalized Linear Models. Authors: Rafael S. de Souza, Alberto Krone-Martins, Jonathan Elliott, Joseph Hilbe.
- CpGFilter** CpG Filtering Method Based on Intra-class Correlation Coefficients. Author: Jun Chen.
- Crossover** Crossover Designs. Author: Kornelius Rohmeyer.
- D2C** Predicting Causal Direction from Dependency Features. Authors: Gianluca Bontempi, Catharina Olsen, Maxime Flauder.
- DAMOCLES** Dynamic Assembly Model Of Colonization, Local Extinction and Speciation. Authors: Rampal S. Etienne and Alex L. Pigot.
- DDIwR** DDI with R. Author: Adrian Dusa [aut, cre].
- DLMtool** Data-Limited Methods Toolkit. Author: Tom Carruthers.
- DOvalidation** Local Linear Hazard Estimation with Do-Validated and Cross-Validated Bandwidths. Authors: M.L. Gamiz, E. Mammen, M.D. Martinez-Miranda and J.P. Nielsen.
- DStree** Recursive Partitioning for Discrete-Time Survival Trees. Authors: Peter Mayer, Denis Larocque, Matthias Schmid. In view: *Survival*.
- DepthProc** Authors: Daniel Kosiorowski, Mateusz Bocian, Anna Wegrzynkiewicz and Zygmunt Zawadzki.
- Deriv** Symbolic Differentiation. Author: Andrew Clausen.
- DetMCD** DetMCD Algorithm (Robust and Deterministic Estimation of Location and Scatter). Authors: Vakili Kaveh [aut, cre], Mia Hubert [ths].

- DynNom** A Dynamic Nomogram for Linear and Generalized Linear Models as Shiny Applications. Authors: Amirhossein Jalali, Alberto Alvarez-Iglesias, John Newell.
- EBglmnet** Empirical Bayesian Lasso and Elastic Net Methods for Generalized Linear Models. Author: Anhui Huang.
- EGRET** Exploration and Graphics for RivEr Trends. Authors: Robert Hirsch [aut], Laura DeCicco [aut, cre].
- EIAdata** R Wrapper for the Energy Information Administration (EIA) API. Author: Matthew Brigida. In view: *WebTechnologies*.
- EMDomics** Earth Mover's Distance for Differential Analysis of Genomics Data. Authors: Daniel Schmolze [aut, cre], Andrew Beck [aut], Sheida Nabavi [aut].
- ENMeval** Automated runs and evaluations of ecological niche models. Authors: Robert Muscarella, Peter J. Galante, Mariano Soley-Guardia, Robert A. Boria, Jamie M. Kass, Maria Uriarte and Robert P. Anderson.
- ENiRG** Ecological Niche in R and GRASS. Authors: Fernando Canovas, Chiara Magliozzi, Jose Antonio Palazon-Ferrando, Frederico Mestre, Mercedes Gonzalez-Wanguemert.
- EcoVirtual** Simulation of Ecological Models. Authors: Alexandre Adalardo de Oliveira and Paulo Inacio Prado.
- EffectTreat** Prediction of Therapeutic Success. Authors: Wim Van der Elst, Ariel Alonso and Geert Molenberghs.
- EnsembleBase** Extensible Package for Parallel, Batch Training of Base Learners for Ensemble Modeling. Authors: Alireza S. Mahani, Mansour T.A. Sharabiani.
- EnsembleCV** Extensible Package for Cross-Validation-Based Integration of Base Learners. Authors: Mansour T.A. Sharabiani, Alireza S. Mahani.
- EnsemblePCReg** Extensible Package for Principal-Component-Regression-based Integration of Base Learners. Authors: Mansour T.A. Sharabiani, Alireza S. Mahani.
- EnsemblePenReg** Extensible Classes and Methods for Penalized-Regression-based Integration of Base Learners. Authors: Mansour T.A. Sharabiani, Alireza S. Mahani.
- EpiDynamics** Dynamic Models in Epidemiology. Authors: Oswaldo Santos Baquero [aut, cre], Fernando Silveira Marques [aut].
- Eplot** Plotting longitudinal series. Author: Eran Raviv.
- FADA** Variable selection for supervised classification in high dimension. Authors: Emeline Perthame, Chloe Friguet and David Causeur.
- FAMILY** A Convex Formulation for Modeling Interactions with Strong Heredity. Author: Asad Haris.
- FDGcopulas** Multivariate Dependence with FDG Copulas. Authors: Gildas Mazo, Stephane Girard.
- FPDclustering** PD-Clustering and Factor PD-Clustering. Authors: Cristina Tortora and Paul D. McNicholas.
- FRESA.CAD** FeatuRE Selection Algorithms for Computer Aided Diagnosis. Author: Jose Gerardo Tamez-Pena.
- FatTailsR** Power Hyperbolic Functions and Kiener Distributions. Author: Patrice Kiener. In view: *Distributions*.
- FeatureHashing** Creates a Model Matrix via Feature Hashing With a Formula Interface. Author: Wush Wu [aut, cre].

- ForIT** Functions from the 2nd Italian Forest Inventory (INFC). Authors: Nicola Puletti, Marco Mura, Cristiano Castaldi, Maurizio Marchi, Ugo Chiavetta, Roberto Scotti.
- ForwardSearch** Forward Search using asymptotic theory. Author: Bent Nielsen.
- Frames2** Estimation in dual frame surveys. Authors: Antonio Arcos, Maria del Mar Rueda, Maria Giovanna Ranalli and David Molina.
- FreeSortR** Free Sorting data analysis. Author: Philippe Courcoux.
- FunctionalNetworks** An algorithm for gene and gene set network inference. Author: Alejandro Quiroz-Zarate.
- GCAI.bias** Guided Correction Approach for Inherited bias (GCAI.bias). Author: Guoshuai Cai.
- GCPM** Generalized Credit Portfolio Model. Author: Kevin Jakob. In view: *Finance*.
- GENLIB** Genealogical Data Analysis. Authors: Louis Houde [aut], Jean-Francois Lefebvre [aut], Valery Roy-Lagace [aut], Sebastien Lemieux [aut], Michael J. Fromberger [ctb], Marie-Helene Roy-Gagnon [cre].
- GLDreg** Fit GLD Regression Model and GLD Quantile Regression Model to Empirical Data. Authors: Steve Su, with contributions from the R Core Team.
- GMDH** Predicting and Forecasting Time Series via GMDH-Type Neural Network Algorithms. Authors: Osman Dag, Ceylan Yozgatligil. In view: *TimeSeries*.
- GPC** Generalized Polynomial Chaos. Authors: Miguel Munoz Zuniga and Jordan Ko.
- GPareto** Gaussian Processes for Pareto Front Estimation and Optimization. Authors: Mickael Binois, Victor Picheny.
- GUILDS** Implementation of sampling formulas for the unified neutral model of biodiversity and biogeography, with or without guild structure. Author: Thijs Janzen.
- GUIProfiler** Profiler Graphical User Interface. Authors: Fernando de Villar and Angel Rubio. In view: *HighPerformanceComputing*.
- GenWin** Spline Based Window Boundaries for Genomic Analyses. Author: Timothy M. Beissinger.
- GeneticSubsetter** Identify Favorable Subsets of Germplasm Collections. Authors: Ryan C. Graebner and Alfonso Cuesta-Marcos.
- GeoDE** A geometrical Approach to Differential expression and gene-set enrichment. Authors: Neil R. Clark and Avi Ma'ayan.
- GlobalOptions** Generate Functions to Get or Set Global Options. Author: Zuguang Gu.
- GrammR** Graphical Representation and Modeling of Metagenomic Reads. Authors: Deepak N. Ayyala, Shili Lin.
- HSSVD** Biclustering with Heterogeneous Variance. Authors: Guanhua Chen [aut, cre], Michael Kosorok [aut], Shannon Holloway [ctb].
- HWxtest** Exact Tests for Hardy-Weinberg Proportions. Author: Bill Engels.
- HiDimMaxStable** Inference on High Dimensional Max-Stable Distributions. Authors: Alexis Bienvenüe [aut, cre], Christian Robert [aut].
- HierO** A graphical user interface for calculating power and sample size for hierarchical data. Author: Kari Tokola.

- IDPSurvival** Imprecise Dirichlet Process for Survival Analysis. Authors: Francesca Mangili, Alessio Benavoli, Cassio P. de Campos, Marco Zaffalon. In view: *Survival*.
- IRTShiny** Item Response Theory via Shiny. Authors: William Kyle Hamilton, Atsushi Mizumoto. In view: *Psychometrics*.
- ISOpureR** Deconvolution of Tumour Profiles. Authors: Gerald Quon [aut], Catalina V Anghel [aut, trl], Syed Haider [aut], Francis Nguyen [aut], Amit G Deshwar [aut], Quaid D Morris [aut], Paul C Boutros [aut, cre].
- InvariantCausalPrediction** Invariant Causal Prediction. Author: Nicolai Meinshausen.
- JAGUAR** Joint Analysis of Genotype and Group-specific Variability Using a Novel Score Test Approach to Map eQTL. Authors: Chaitanya R. Acharya and Andrew S. Allen.
- JMdesign** Joint Modeling of Longitudinal and Survival Data – Power Calculation. Authors: Emil A. Cornea, Liddy M. Chen, Bahjat F. Qaqish, Haitao Chu, and Joseph G. Ibrahim.
- KODAMA** Knowledge discovery by accuracy maximization. Authors: Stefano Cacciatore, Claudio Luchinat, Leonardo Tenori.
- LDRTools** Tools for Linear Dimension Reduction. Authors: Eero Liski [aut, cre], Klaus Nordhausen [aut], Hannu Oja [aut], Anne Ruiz-Gazen [aut].
- LakeMetabolizer** Tools for the analysis of ecosystem metabolism. Authors: Luke Winslow, Jake Zwart, Ryan Batt, Jessica Corman, Hilary Dugan, Paul Hanson, Gordon Holtgrieve, Aline Jaimes, Jordan Read, Richard Woolway.
- LinCal** Static Univariate Frequentist and Bayesian Linear Calibration. Authors: Derick L. Rivers and Edward L. Boone.
- LinRegInteractive** Interactive Interpretation of Linear Regression Models. Author: Martin Meermeyer. In view: *Econometrics*.
- LindenmayerR** Functions to Explore L-Systems (Lindenmayer Systems). Author: Bryan Hanson [aut, cre].
- MATA** Model-Averaged Tail Area Wald (MATA-Wald) Confidence Interval. Author: Daniel Turek [aut, cre].
- MAVIS** Meta Analysis via Shiny. Authors: William Kyle Hamilton, Atsushi Mizumoto. In view: *MetaAnalysis*.
- MCAvariants** Multiple Correspondence Analysis Variants. Author: Rosaria Lombardo.
- MCS** Model Confidence Set Procedure. Authors: Leopoldo Catania and Mauro Bernardi.
- MGL** Module Graphical Lasso. Author: Safiye Celik.
- MGRASter** API Client for the MG-RAST Server of the US DOE KBase. Author: Daniel T. Braithwaite [aut, cre].
- MGSDA** Multi-Group Sparse Discriminant Analysis. Author: Irina Gaynanova.
- MOrder** Check Time Homogeneity and Markov Chain Order. Authors: Akshay Chougule, Robert Canales.
- MPAgenomics** Multi-Patient Analysis of Genomic Markers. Authors: Quentin Grimonprez with contributions from Guillemette Marot and Samuel Blanck. Some functions use code created by Sjoerd Vosse, Mark van de Wiel, Pierre Neuvial, Henrik Bengtsson.
- MRH** Multi-Resolution Estimation of the Hazard Rate. Authors: Yolanda Hagar, Yuanting Chen, Vanja Dukic.

- MRSP** Multinomial Response Models with Structured Penalties. Author: Wolfgang Poessnecker.
- MSIseq** Assess Tumor Microsatellite Instability with a Decision Tree Classifier from Exome Somatic Mutations. Author: Mini Huang.
- MST** Multivariate Survival Trees. Authors: Xiaogang Su, Peter Calhoun, and Juanjuan Fan. In view: *Survival*.
- MVar.pt** Analise Multivariada (Brazilian Portuguese). Authors: Paulo Cesar Ossani and Marcelo Angelo Cirillo.
- MareyMap** Estimation of meiotic recombination rates using marey maps. Authors: Aurelie Siberchicot, Clement Rezvoy, Delphine Charif, Laurent Gueguen and Gabriel Marais.
- MatchingFrontier** Computation of the Balance Sample Size Frontier in Matching Methods for Causal Inference. Authors: Gary King, Christopher Lucas, and Richard Nielsen.
- MenuCollection** Collection of Configurable GTK+ Menus. Author: Gianmarco Polotti.
- MetFns** Analysis of Visual Meteor Data. Author: Kristina Veljkovic.
- MetaLandSim** Metapopulation and Landscape Simulation. Authors: Frederico Mestre, Fernando Canovas, Ricardo Pita, Antonio Mira, Pedro Beja.
- MfUSampler** Multivariate-from-Univariate (MfU) MCMC Sampler. Authors: Alireza S. Mahani, Mansour T.A. Sharabiani.
- MixGHD** Model Based Clustering, Classification and Discriminant Analysis Using the Mixture of Generalized Hyperbolic Distributions. Authors: Cristina Tortora, Ryan P. Browne, Brian C. Franczak and Paul D. McNicholas.
- MixedTS** Mixed Tempered Stable Distribution. Authors: Lorenzo Mercuri, Edit Rroji.
- MultNonParam** Multivariate Nonparametric Methods. Authors: John E. Kolassa and Stephane Jankowski.
- MultiSV** Identification of structural variations in multiple populations based on whole genome resequencing. Author: Khurram Maqbool.
- NB** Maximum Likelihood method in estimating effective population size from genetic data. Author: Tin-Yu Hui.
- NHANES** NHANES data. Authors: Randall Pruim, Daniel Kaplan, Nicholas Horton.
- NHMM** Bayesian NHMM Modeling (Multiple Time Series). Author: Tracy Holsclaw.
- NISTunits** Fundamental Physical Constants and Unit Conversions from NIST. Author: Jose Gama [aut, cre].
- NORTARA** Generation of Multivariate Data with Arbitrary Marginals. Author: Po Su [aut, cre].
- NPBayesImpute** Non-parametric Bayesian Multiple Imputation for Categorical Data. Authors: Quanli Wang, Daniel Manrique-Vallier, Jerome P. Reiter and Jingchen Hu.
- NeuralNetTools** Visualization and Analysis Tools for Neural Networks. Author: Marcus W. Beck [aut, cre].
- ONETr** Efficient authenticated interaction with the O*NET API. Author: Eric Knudsen. In view: *WebTechnologies*.
- OptionPricing** Option Pricing with Efficient Simulation Algorithms. Authors: Kemal Dingec, Wolfgang Hormann. In view: *Finance*.

- OutrankingTools** Functions for Solving Multiple-criteria Decision-making Problems. Author: Michel Prombo.
- P2C2M** Posterior Predictive Checks of Coalescent Models. Authors: Michael Gruenstaedl, Noah Reid.
- PANDA** Preferential Attachment Based Common Neighbor Distribution Derived Functional Associations. Authors: Hua Li and Pan Tong.
- PANICr** PANIC Tests of Nonstationarity. Author: Steve Bronder.
- PASWR2** Probability and Statistics with R, Second Edition. Author: Alan T. Arnholt.
- PGICA** Parallel Group ICA Algorithm. Authors: Shaojie Chen, Lei Huang, Huitong Qiu, Ani Eloyan, Brian Caffo and Ciprian Crainiceanu. In view: *HighPerformanceComputing*.
- PHENIX** Phenotypic Integration Index. Authors: R. Torices, A. J. Muñoz-Pajares.
- PLSbiplot1** The Partial Least Squares (PLS) Biplot. Authors: Opeoluwa F. Oyedele [aut, cre], Sugnet Gardner-Lubbe [aut].
- PLordprob** Multivariate Ordered Probit Model via Pairwise Likelihood. Authors: Euloge Clovis Kenne Pagui [aut, cre], Antonio Canale [aut], Alan Genz [ctb], Adelchi Azzalini [ctb].
- PRROC** Precision-Recall and ROC Curves for Weighted and Unweighted Data. Authors: Jan Grau and Jens Keilwagen.
- ParallelForest** Random Forest Classification with Parallel Computing. Author: Bertram Jeong [aut, cre, cph].
- PepPrep** Insilico peptide mutation, digestion and homologous comparison. Author: Rafael Dellen.
- PerFit** Person Fit. Author: Jorge N. Tendeiro.
- PhyActBedRest** Marks periods of sleep in Actigraph accelerometer data. Authors: J. Dustin Tracy, Zhiyi Xu, Leena Choi, Sari Acra, Kong Y. Chen, Maciej S. Buchowski.
- PolyPatEx** Paternity exclusion in autopolyploid species. Author: Alexander Zwart.
- PrevMap** Geostatistical Modelling of Spatially Referenced Prevalence Data. Authors: Emanuele Giorgi, Peter J. Diggle.
- QoLR** Analysis of Health-Related Quality of Life in oncology. Author: Amelie Anota.
- QualInt** Test for Qualitative Interactions. Authors: Lixi Yu, Eun-Young Suh, Guohua (James) Pan.
- Quor** Quantile Ordering. Authors: Adriano Polpo, Carlos A. de B. Pereira, Cassio P. de Campos.
- R6** Classes with reference semantics. Author: Winston Chang [aut, cre].
- RAM** R for Amplicon-based Metagenomics. Authors: Wen Chen and Joshua Simpson.
- RAMP** Regularized Generalized Linear Models with Interaction Effects. Authors: Yang Feng, Ning Hao and Hao Helen Zhang.
- RAdwords** Loading Google Adwords Data Into R. Authors: Johannes Burkhardt, Matthias Bannert.
- RCMIP5** Tools for Manipulating and Summarizing CMIP5 Data. Authors: Ben Bond-Lamberty [aut], Kathe Todd-Brown [aut, cre].

- RConics** Computations on Conics. Author: Emanuel Huber.
- RCryptsy** Access to Cryptsy Crypto-Currency Exchange Public Information API via R. Author: William Kyle Hamilton. In view: *WebTechnologies*.
- RDML** Importing real-time thermo cyler (qPCR) data from RDML format files. Authors: Konstantin A. Blagodatskikh [cre, aut], Stefan Roediger [aut], Michal Burdukiewicz [aut].
- RDataCanvas** Basic Runtime Support for Datacanvas.io. Author: Xiaolin Zhang [aut, cre]. In view: *WebTechnologies*.
- RECA** Relevant Component Analysis for Supervised Distance Metric Learning. Author: Nan Xiao.
- REDCapR** Interaction between R and REDCap. Authors: Will Beasley [aut, cre], David Bard [ctb], Thomas Wilson [ctb], John J Aponte [ctb], Rollie Parrish [ctb], Benjamin Nutter [ctb], Andrew Peters [ctb].
- RGGA** A Google Analytics API client for R. Authors: Artem Klevtsov [aut, cre], Philipp Upravitelev [ctb], Olga Shramko [ctb]. In view: *WebTechnologies*.
- RGoogleAnalytics** R Wrapper for the Google Analytics API. Authors: Michael Pearmain. Contributions from Nick Mihailowski, Vignesh Prajapati, Kushan Shah and Nicolas Remy. In view: *WebTechnologies*.
- RJSDMX** R Interface to SDMX Web Services. Authors: Attilio Mattiocco, Diana Nicoletti, Gianpaolo Lopez. In view: *WebTechnologies*.
- RMKdiscrete** Sundry Discrete Probability Distributions. Author: Robert M. Kirkpatrick.
- RMOA** Connect R with MOA for Massive Online Analysis. Authors: Jan Wijffels [aut, cre], BNOSAC [cph].
- RMOAjars** External jars required for package **RMOA**. Author: See file AUTHORS.
- RMRAINGEN** R Multi-site RAINfall GENEretor: generate daily time series of rainfall from monthly mean values. Author: Emanuele Cordano.
- RNeXML** Implement semantically rich I/O for the NeXML format. Authors: Carl Boettiger [cre, aut], Scott Chamberlain [aut], Hilmar Lapp [aut], Kseniia Shumelchik [aut], Rutger Vos [aut]. In view: *Phylogenetics*.
- RODBCext** Parameterized queries extension for **RODBC**. Authors: Mateusz Zoltak [aut, cre], Brian Ripley [aut], Michael Lapsley [aut].
- RPANDA** Phylogenetic ANalyses of DiversificAtion. Author: H el ene Morlon [aut, cre, cph].
- RRreg** Correlation and Regression Analyses for Randomized Response Data. Authors: Daniel W. Heck [aut, cre], Morten Moshagen [aut].
- RSQLServer** SQL Server R Database Interface (DBI). Authors: Imanuel Costigan [aut, cre], The jTDS Project [aut] (for MSSQL Server driver), Simon Urbanek [ctb], The Legion Of The Bouncy Castle [cph, ctb].
- RStars** Access to the Digital Universe Data set API. Author: William Kyle Hamilton. In view: *WebTechnologies*.
- RSurveillance** Design and Analysis of Disease Surveillance Activities. Author: Evan Sergeant.
- RTDE** Robust Tail Dependence Estimation. Authors: Christophe Dutang [aut, cre], Armelle Guillou [ctb], Yuri Goegebeur [ctb]. In view: *Distributions*.

- RTriangle** A 2D Quality Mesh Generator and Delaunay Triangulator. Authors: Jonathan Shewchuk, David C. Sterratt.
- Ramd** Tools For Managing File/function Dependencies In R. Author: Robert Krzyzanowski.
- RcmdrMisc** R Commander Miscellaneous Functions. Authors: John Fox [aut, cre], Robert Muenchen [ctb], Dan Putler [ctb].
- RcmdrPlugin.EcoVirtual Rcmdr** EcoVirtual Plugin. Authors: Alexandre Adalardo de Oliveira and Paulo Inacio Prado.
- RcmdrPlugin.RMTCJags** R MTC Jags **Rcmdr** Plugin. Author: Marcelo Goulart Correia.
- RcmdrPlugin.ROC Rcmdr** Receiver Operator Characteristic Plug-In package. Authors: Daniel-Corneliu Leucuta [aut, cre], Mihaela Hedesiu [ctb], Andrei Achimas [ctb], Oana Almasan [ctb].
- RcppAnnoy Rcpp** Bindings for Annoy, a Library for Approximate Nearest Neighbors. Author: Dirk Eddelbuettel.
- RcppDL** Deep Learning Methods via **Rcpp**. Authors: Qiang Kou, Yusuke Sugomori.
- RcppMLPACK Rcpp** Integration for MLPACK Library. Authors: Qiang Kou, Ryan Curtin.
- RcppParallel** Parallel programming tools for **Rcpp**. Authors: JJ Allaire; Romain Francois; Intel, Inc.; Marcus Geelnard.
- Rdsdp** R Interface to DSDP Semidefinite Programming Library. Authors: Zhisu Zhu, Yinyu Ye (DSDP by Steve Benson, Yinyu Ye and Xiong Zhang).
- RealVAMS** Multivariate VAM Fitting. Authors: Andrew Karl, Jennifer Broatch, and Jennifer Green.
- RegressionFactory** Expander Functions for Generating Full Gradient and Hessian from Single- and Multi-Slot Base Distributions. Authors: Alireza S. Mahani, Mansour T.A. Sharabiani.
- ReorderCluster** Reordering the dendrogram according to the class labels. Authors: Natalia Novoselova, Frank Klawonn, Junxi Wang.
- RmixmodCombi** Combining Mixture Components for Clustering. Authors: J.-P. Baudry and G. Celeux.
- RootsExtremaInflections** Finds roots, extrema and inflection points of a curve. Author: Demetris T. Christopoulos. In view: *NumericalMathematics*.
- RoughSetKnowledgeReduction** Simplification of Decision Tables using Rough Sets. Author: Alber Sanchez.
- SBRect** Detecting structural breaks using rectangle covering (non-parametric method). Authors: Paul Fischer [aut, cre, cph], Astrid Hilbert [ctb, cph].
- SCEPtERbinary** Stellar CharactEristics Pisa Estimation gRid for Binary Systems. Authors: Matteo Dell'Omodarme [aut, cre], Giada Valle [aut].
- SEAsic** Score Equity Assessment summary index computation. Authors: Anne Corinne Huggins-Manley [aut, cre], Douglas Whitaker [aut].
- SOD** SOD for multidimensional scaling. Author: Martin Jakt.
- SOR** Estimation using Sequential Offsetted Regression. Authors: Lee S. McDaniel, Jonathan S. Schildcrout.
- SPREDA** Statistical Package for Reliability Data Analysis. Authors: Yili Hong, Yimeng Xie, and Zhibing Xu.

- SQDA** Sparse Quadratic Discriminant Analysis. Author: Jiehuan Sun.
- SSrat** Two-dimensional sociometric status determination with rating scales. Author: Hans Landsheer.
- ScoreGGUM** Score Persons Using the Generalized Graded Unfolding Model. Authors: David R. King and James S. Roberts.
- SelvarMix** Regularization for variable selection in model-based clustering and discriminant analysis. Authors: Mohammed Sedki, Gilles Celeux, Cathy Maugis-Rabusseau.
- SimplicialCubature** Numerical integration of functions over simplices. Authors: John P. Nolan, with parts based on code by Alan Genz. In view: *NumericalMathematics*.
- SoundexBR** Phonetic-Coding For Portuguese. Author: Daniel Marcelino.
- SphericalK** Spherical K-function. Authors: Scott Robeson, Ao Li, Chunfeng Huang.
- StatMethRank** Statistical Methods for Ranking Data. Author: Li Qinglong.
- Statomica** Statomica utility package. Authors: Zahra Montazeri, Alaa Ali, Kyle Leckett, Marta Padilla and David R. Bickel.
- TDA** Statistical Tools for Topological Data Analysis. Authors: Brittany T. Fasy, Jisu Kim, Fabrizio Lecci, Clement Maria.
- TDboost** A Boosted Nonparametric Tweedie Model. Authors: Yi Yang, Wei Qian, Hui Zou.
- TEEReg** Trimmed Elemental Estimation for Linear Models. Authors: Wei Jiang and Matthew S. Mayo. In view: *Robust*.
- TR8** A tool for downloading functional traits data for plant species. Author: Gionata Bocci. In view: *WebTechnologies*.
- TRSbook** Functions and Datasets to Accompany the Book “The R Software: Fundamentals of Programming and Statistical Analysis”. Authors: Pierre Lafaye de Micheaux, Remy Drouilhet, Benoit Liqueur.
- TableMonster** Table Monster. Author: Grant Izmirlian Jr.
- UPMASK** Unsupervised Photometric Membership Assignment in Stellar Clusters. Authors: Alberto Krone-Martins, Andre Moitinho.
- USAboundaries** Historical Boundaries of the United States of America, 1629–2000. Authors: Lincoln Mullen [aut, cre], Dr. William M. Scholl Center for American History and Culture, The Newberry Library [cph].
- V8** Embedded JavaScript Engine. Author: Jeroen Ooms.
- W3CMarkupValidator** R Interface to W3C Markup Validation Services. Author: Kurt Hornik [aut, cre]. In view: *WebTechnologies*.
- WRS2** Wilcox Robust Estimation and Testing. Authors: Patrick Mair [cre, aut], Felix Schoenbrodt [aut], Rand Wilcox [aut]. In view: *Robust*.
- Watersheds** Spatial Watershed Aggregation and Spatial Drainage Network Analysis. Author: J.A. Torres.
- WaveletComp** Computational Wavelet Analysis. Authors: Angi Roesch and Harald Schmidbauer.
- WhiteStripe** Whitestripe White Matter Normalization for Magnetic Resonance Images. Authors: Taki Shinohara, John Muschelli.

- aRxiv** Interface to the arXiv API. Authors: Karthik Ram [aut], Karl Broman [aut, cre]. In view: *WebTechnologies*.
- acid** Analysing Conditional Distributions of Income. Author: Alexander Sohn.
- acnr** Annotated Copy-Number Regions. Authors: Morgane Pierre-Jean and Pierre Neuvial.
- acp** Autoregressive Conditional Poisson. Author: Siakoulis Vasileios.
- activity** Animal Activity Statistics. Author: Marcus Rowcliffe.
- adaptDA** Adaptive Mixture Discriminant Analysis. Author: Charles Bouveyron.
- addreg** Additive Regression for Discrete Data. Author: Mark Donoghoe.
- adwave** Wavelet Analysis of Genomic Data from Admixed Populations. Author: Jean Sanderson.
- alleHap** Allele simulation and Haplotype reconstruction from pedigree databases. Authors: Nathan Medina-Rodriguez and Angelo Santana.
- analogueExtra** Additional functions for use with the **analogue** package. Author: Gavin L. Simpson [aut, cre].
- anim.plots** Simple Animated Plots For R. Author: David Hugh-Jones.
- aoos** Another Object Orientation System. Author: Sebastian Warnholz.
- apc** Age-period-cohort analysis. Author: Bent Nielsen.
- aplore3** Datasets from Hosmer, Lemeshow and Sturdivant, “Applied Logistic Regression” (3rd ed.). Author: Luca Braglia [aut, cre].
- aprean3** Datasets from Draper and Smith “Applied Regression Analysis” (3rd ed., 1998). Author: Luca Braglia [aut, cre].
- archdata** Example Datasets from Archaeological Research. Author: David L. Carlson.
- archiDART** Plant Root System Architecture Analysis Using DART Output Files. Authors: Benjamin M Delory, Caroline Baudson, Yves Brostaux, Patrick du Jardin, Loic Pages, Pierre Delaplace.
- archivist** Tools for Storing, Restoring and Searching for R Objects. Authors: Przemyslaw Biecek [aut, cre], Marcin Kosinski [aut].
- astrodatR** Astronomical Data. Author: Eric D. Feigelson.
- astrolibR** Astronomy Users Library. Authors: Arnab Chakraborty and Eric D. Feigelson.
- atmcmc** Automatically Tuned Markov Chain Monte Carlo. Author: Jinyoung Yang.
- attribrisk** Population Attributable Risk. Authors: Louis Schenck, Elizabeth Atkinson, Cynthia Crowson, Terry Therneau.
- babynames** US baby names 1880–2013. Author: Hadley Wickham [aut, cre].
- bamboo** Protein Secondary Structure Prediction Using the Bamboo Method. Author: David B. Dahl.
- bayou** Bayesian fitting of Ornstein-Uhlenbeck models to phylogenies. Authors: Josef C. Uyeda, Jon Eastman and Luke Harmon.
- bde** Bounded Density Estimation. Authors: Guzman Santafo, Borja Calvo, Aritz Perez and Jose A. Lozano.
- bdscale** Remove Weekends and Holidays From **ggplot2** Axes. Author: Dave Mills.

- bglm** Bayesian Estimation in Generalized Linear Models. Authors: Nicolas Molano-Gonzalez, Edilberto Cepeda-Cuervo.
- binequality** Methods for Analyzing Binned Income Data. Authors: Samuel V. Scarpino, Paul von Hippel, and Igor Holas.
- binr** Cut Numeric Values Into Evenly Distributed Groups. Author: Sergei Izrailev.
- bio3d** Biological Structure Analysis. Authors: Barry Grant, Xin-Qiu Yao, Lars Skjaerven, Julien Ide.
- biogram** N-Gram Analysis of Biological Sequences. Authors: Michal Burdukiewicz [cre, aut], Piotr Sobczyk [aut], Chris Lauber [aut].
- blocksdesign** Nested Block Designs for Unstructured Treatments. Author: R. N. Edmondson.
- blsAPI** Request Data From The U.S. Bureau of Labor Statistics API. Author: Michael Silva.
- bnormnlr** Bayesian Estimation for Normal Heteroscedastic Nonlinear Regression Models. Authors: Nicolas Molano-Gonzalez, Marta Corrales Bossio, Maria Fernanda Zarate, Edilberto Cepeda-Cuervo.
- bootSVD** Fast, Exact Bootstrap Principal Component Analysis for High Dimensional Data. Author: Aaron Fisher.
- boral** Bayesian Ordination and Regression AnaLysis. Author: Francis K.C. Hui.
- breakaway** Species richness estimation and modelling. Authors: Amy Willis and John Bunge.
- broom** Convert Statistical Analysis Objects Into Tidy Data Frames. Authors: David Robinson [aut, cre], Matthieu Gomez [ctb], Boris Demeshev [ctb], Hadley Wickham [ctb].
- btf** Estimates univariate function via Bayesian trend filtering. Author: Edward A. Roualdes.
- bvarsv** Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters. Author: Fabian Krueger.
- causaleffect** Deriving Expressions of Joint Interventional Distributions in Causal Models. Author: Santtu Tikka.
- cdcsis** Conditional Distance Correlation and Its Related Feature Screening Method. Authors: Canhong Wen, Wenliang Pan, Mian Huang, and Xueqin Wang.
- cents** Censored time series. Authors: A.I. McLeod, Nagham M. Mohammad, Justin Veenstra and Abdel El-Shaarawi.
- checkpoint** Install Packages from Snapshots on the Checkpoint Server for Reproducibility. Author: Revolution Analytics.
- childsds** Calculation of standard deviation scores adduced from different growth standards. Author: Mandy Vogel [aut, cre].
- choiceDes** Design Functions for Choice Studies. Author: Jack Horne [aut, cre].
- choroplethrAdmin1** Contains an Administrative-Level-1 Map of the World. Author: Ari Lamstein.
- choroplethrMaps** Contains maps used by the **choroplethr** package. Author: Ari Lamstein.
- cjoint** AMCE Estimator for Conjoint Experiments. Authors: Anton Strezhnev, Jens Hainmueller, Daniel Hopkins, Teppei Yamamoto.

- classyfire** Robust multivariate classification using highly optimised SVM ensembles. Authors: Eleni Chatzimichali and Conrad Bessant.
- cmprskQR** Analysis of Competing Risks Using Quantile Regressions. Authors: Stephan Dlugosz, based on code from Limin Peng and Ruosha Li.
- cmvnorm** Complex multivariate Gaussian distribution. Author: Robin K. S. Hankin.
- coenocliner** Coenocline simulation. Authors: Gavin L. Simpson [aut, cre], Francisco Rodriguez-Sanchez [ctb].
- commentr** Print Nicely Formatted Comments for use in Script Files. Author: Erik Bulow.
- compendiumdb** Tools for Storage and Retrieval of Gene Expression Data. Authors: Umesh Nandal, Perry D. Moerland.
- complmrob** Robust Linear Regression with Compositional Data as Covariates. Author: David Kepplinger.
- confidence** Confidence Estimation of Environmental State Classifications. Authors: Willem M. G. M. van Loon [aut, cph], Dennis J. J. Walvoort [aut, cre].
- cooptrees** Cooperative aspects of optimal trees in weighted graphs. Author: Manuel Fontenla.
- copCAR** Fitting the copCAR regression model for discrete areal data. Authors: Emily Goren and John Hughes.
- coreTDT** TDT for compound heterozygous and recessive models. Authors: Yu Jiang, Andrew S Allen.
- cosinor** Tools for estimating and predicting the cosinor model. Author: Michael Sachs.
- couchDB** Connect and work with couchDB databases. Author: Aleksander Dietrichson.
- coxinterval** Cox-type models for interval-censored data. Authors: Audrey Boruvka and Richard J. Cook. In view: *Survival*.
- cqrReg** Quantile, Composite Quantile Regression and Regularized Versions. Authors: Jueyu Gao and Linglong Kong.
- crayon** Colored Terminal Output. Author: Gabor Csardi [aut, cre].
- crossReg** Confidence intervals for crossover points of two simple regression lines. Author: Sunbok Lee.
- curl** A Connection Interface to Libcurl. Author: Jeroen Ooms. In view: *WebTechnologies*.
- dashboard** Interactive Data Visualization with D3.js. Author: Johann Laurent.
- dataRetrieval** Retrieval Functions for USGS and EPA Hydrologic and Water Quality Data. Authors: Robert Hirsch [aut], Laura DeCicco [aut, cre], David Lorenz [aut]. In view: *WebTechnologies*.
- db.r** A Database Exploration Tool with Assisted Querying and Schema Exploration. Author: Greg Lamp.
- dbarts** Discrete Bayesian Additive Regression Trees Sampler. Authors: Hugh Chipman, Robert McCulloch, Vincent Dorie.
- dcGOR** Analysis of ontologies and protein domain annotations. Authors: Hai Fang and Julian Gough.
- dcmr** Attribute profile estimation using Diagnostic Classification Models and MCMC. Authors: Diane Losardo and Margi Dubal.

- decompr** Export Decomposition (Wang-Wei-Zhu and source). Authors: Bastiaan Quast [aut, cre], Fei Wang [aut], Victor Kummritz [aut].
- deming** Deming, Thiel-Sen and Passing-Bablok Regression. Author: Terry Therneau.
- densityClust** Clustering by fast search and find of density peaks. Author: Thomas Lin Pedersen.
- descomponer** Seasonal Adjustment by Frequency Analysis. Author: Francisco Parra.
- dfcomb** Phase I/II Adaptive Dose-Finding Design for Combination Studies. Authors: Marie-Karelle Riviere and Jacques-Henri Jourdan.
- dfmta** Phase I/II Adaptive Dose-Finding Design For MTA. Authors: Marie-Karelle Riviere and Jacques-Henri Jourdan.
- diffeR** Difference Metrics for Comparing Pairs of Maps. Authors: Robert Gilmore Pontius Jr., Alí Santacruz.
- digitalPCR** Estimate copy number for digital PCR. Author: Xutao Deng.
- disposables** Create Disposable R Packages for Testing. Author: Gabor Csardi [aut, cre].
- documair** Automatic Documentation for R packages. Authors: Jean-Baptiste Denis, Regis Pouillot, Kien Kieu.
- dotenv** Load environment variables from '.env'. Author: Gabor Csardi [aut, cre].
- dslice** Dynamic slicing. Authors: Chao Ye and Bo Jiang.
- dummy** Automatic Creation of Dummies with Support for Predictive Modeling. Authors: Michel Ballings and Dirk Van den Poel.
- dunn.test** Dunn's Test Of Multiple Comparisons Using Rank Sums. Author: Alexis Dinno.
- dygraphs** Interface to Dygraphs Interactive Time Series Charting Library. Authors: Dan Vanderkam [aut, cph] (dygraphs library), JJ Allaire [aut, cre] (R interface), RStudio [cph].
- ecipex** Efficient calculation of fine structure isotope patterns via Fourier transforms of simplex-based elemental models. Author: Andreas Ipsen.
- ecoretriever** R Interface to the EcoData Retriever. Authors: Daniel McGlinn [aut, cre], Ethan White [aut]. In view: *WebTechnologies*.
- ecospat** Spatial ecology miscellaneous methods. Authors: Olivier Broenniman, Blaise Petitpierre, Christophe Randin, Robin Engler, Frank Breiner, Manuela D'Amen, Loic Pellissier, Julien Pottier, Dorothea Pio, Ruben Garcia Mateo, Valeria Di Cola, Wim Hordijk, Anne Dubuis, Daniel Scherrer, Nicolas Salamin and Antoine Guisan.
- ecoval** Procedures for Ecological Assessment of Surface Waters. Authors: Nele Schuwirth and Peter Reichert with contributions by Simone Langhans.
- edgeRun** More Powerful Unconditional Testing of Negative Binomial Means for Digital Gene Expression Data. Author: Emmanuel Dimont.
- eegkit** Toolkit for Electroencephalography Data. Author: Nathaniel E. Helwig.
- eegkitdata** Data for package **eegkit**. Author: Nathaniel E. Helwig.
- ega** Error Grid Analysis. Author: Daniel Schmolze [aut, cre].
- embryogrowth** Tools to Analyze the Thermal Reaction Norm of Embryo Growth. Author: Marc Girondot.

- emil** Evaluation of Modeling without Information Leakage. Authors: Christofer Backlin, Mats Gustafsson.
- empiricalFDR.DESeq2** Simulation-based False Discovery Rate in RNA-seq. Author: Mikhail V. Matz.
- enigma** R Client for the Enigma API. Author: Scott Chamberlain [aut, cre]. In view: *WebTechnologies*.
- enpls** Ensemble Partial Least Squares (EnPLS) Regression. Authors: Nan Xiao, Dong-Sheng Cao, Qing-Song Xu.
- ensurer** Ensure values at runtime. Author: Stefan Holst Milton Bache.
- enviPick** Peak picking for high resolution mass spectrometry data. Author: Martin Loos.
- epi2loc** Epistatic and penetrance models for two-locus genetic interactions. Authors: Raymond Walters, Charles Laurin, Gitta Lubke.
- episplineDensity** Density Estimation with Soft Information by Exponential Epi-splines. Authors: Sam Buttrey, Johannes Roynet, and Roger Wets, based on the Matlab code of Roynet and Wets.
- exCon** Interactive Exploration of Contour Data. Authors: Bryan Hanson [aut, cre], Kristina Mulry [ctb].
- extlasso** Maximum penalized likelihood estimation with extended lasso penalty. Authors: B N Mandal and Jun Ma.
- falsy** Define truthy and falsy values. Author: Gabor Csardi [aut, cre].
- filenamer** Easy Management of File Names. Author: David J. H. Shih.
- flam** Fits Piecewise Constant Models with Data-Adaptive Knots. Author: Ashley Petersen.
- forestplot** Advanced Forest Plot Using grid Graphics. Authors: Max Gordon [aut, cre], Thomas Lumley [aut, ctb].
- frmhet** Regression Analysis of Fractional Responses Under Unobserved Heterogeneity. Author: Joaquim J.S. Ramalho.
- fueleconomy** EPA fuel economy data. Author: Hadley Wickham [aut, cre].
- funFEM** Clustering in the Discriminative Functional Subspace. Author: Charles Bouveyron. In view: *Cluster*.
- funHDDC** Model-based clustering in group-specific functional subspaces. Authors: C. Bouveyron and J. Jacques. In view: *Cluster*.
- fuzzyMM** Map Matching Using Fuzzy Logic. Author: Nikolai Gorte.
- gCat** Graph-based two-sample tests for categorical data. Authors: Hao Chen and Nancy R. Zhang.
- gapmap** Functions for Drawing Gapped Cluster Heatmap with **ggplot2**. Author: Ryo Sakai.
- geesmv** Modified Variance Estimators for Generalized Estimating Equations. Author: Ming Wang.
- gemmR** General Monotone Model. Authors: Jeffrey Chrabaszcz [aut, cre], Joe Tidwell [aut], Michael Dougherty [aut].
- gender** Predict Gender from Names Using Historical Data. Authors: Lincoln Mullen [aut, cre], Cameron Blevins [ctb], Ben Schmidt [ctb].

- genpathmox** Generalized PATHMOX Algorithm for PLS-PM, LS and LAD Regression. Author: Giuseppe Lamberti [aut, cre].
- geoBayes** Analysis of geostatistical data using Bayes and empirical Bayes methods. Authors: Evangelos Evangelou, Vivekananda Roy.
- geocodeHERE** Wrapper for Nokia's HERE Geocoding API. Author: Cory Nissen [aut, cre].
- gets** General-to-Specific (GETS) Modelling and Indicator Saturation Methods. Authors: Felix Pretis, James Reade, Genaro Sucarrat. In view: *TimeSeries*.
- ggRandomForests** Visually Exploring Random Forests. Author: John Ehrlinger.
- ggswissmaps** Offers Various Swiss Maps as **ggplot2** Objects. Author: Sandro Petrillo Burri.
- gimme** Group Iterative Multiple Model Estimation. Author: Stephanie Lane.
- glmvsd** Variable Selection Deviation (VSD) Measures and Instability Tests for High-dimensional Generalized Linear Models. Authors: Ying Nan, Yi Yang, Yuhong Yang.
- gmailr** Access the Gmail RESTful API. Author: Jim Hester. In view: *WebTechnologies*.
- gridGraphics** Redraw Base Graphics Using grid Graphics. Author: Paul Murrell [cre, aut].
- growfunctions** Bayesian non-parametric dependent models for time-indexed functional data. Author: Terrance Savitsky.
- hcp** Change Point Estimation for Regression with Heteroscedastic Data. Authors: Stephen J. Ganocy, Jiayang Sun, and Yulei Wang.
- heritability** Marker-based Estimation of Heritability Using Individual Plant or Plot Data. Authors: Willem Kruijer, with a contribution from Ian White. Contains data collected by Padraic Flood and Rik Kooke.
- hermite** Generalized Hermite Distribution. Authors: David Moriña, Manuel Higuera and Pedro Puig.
- hglm.data** Data for The **hglm** Package. Authors: Xia Shen, Moudud Alam, Lars Ronnegard.
- highD2pop** Two-Sample Tests for Equality of Means in High Dimension. Author: Karl Gregory.
- highTtest** Simultaneous Critical Values for *t*-Tests in Very High Dimensions. Authors: Hongyuan Cao [aut], Michael Kosorok [aut], Shannon Holloway [aut, cre].
- hillmakeR** Perform occupancy analysis. Author: David Gilinson.
- historydata** Data Sets for Historians. Author: Lincoln Mullen [aut, cre].
- hnp** Half-Normal Plots with Simulation Envelopes. Authors: Rafael de Andrade Moral [aut, cre], John Hinde [aut], Clarice Garcia Borges Demetrio [aut].
- hot.deck** Multiple Hot-deck Imputation. Authors: Skyler Cranmer, Jeff Gill, Natalie Jackson, Andreas Murr, Dave Armstrong.
- hpcwld** High Performance Cluster Models Based on Kiefer-Wolfowitz Recursion. Author: Alexander Rumyantsev [aut, cre].
- htmlTable** Advanced Tables for Markdown/HTML. Author: Max Gordon.
- htmlwidgets** HTML Widgets for R. Authors: Ramnath Vaidyanathan, Joe Cheng, JJ Allaire, Yihui Xie, and Kenton Russell.
- hydrostats** Hydrologic indices for daily time series data. Author: Nick Bond.

- iBATCGH** Integrative Bayesian Analysis of Transcriptomic and CGH data. Author: Alberto Cassese.
- iFes** Incremental Feature Selection Algorithm accelerated by GPU. Authors: Qinghan Meng, Fengfeng Zhou.
- ibeemd** Irregular-lattice based ensemble empirical mode decomposition. Author: Maogui Hu.
- ibelief** Implementing Belief Functions. Author: Kuang Zhou; Arnaud Martin.
- ibmdbR** IBM In-Database Analytics for R. Author: IBM Corporation.
- ica** Independent Component Analysis. Author: Nathaniel E. Helwig.
- icamix** Estimation of ICA Mixture Models. Authors: Xiaotian Zhu, David R. Hunter.
- iccbeta** Multilevel model intraclass correlation for slope heterogeneity. Authors: Steven Andrew Culpepper [aut, cph, cre], Herman Aguinis [aut, cph].
- iki.dataclim** Consistency, Homogeneity and Summary Statistics of Climatological Data. Author: Boris Orlovsky.
- ilc** Lee-Carter Mortality Models using Iterative Fitting Algorithms. Authors: Zoltan Butt, Steven Haberman and Han Lin Shang.
- imputeLCMD** A collection of methods for left-censored missing data imputation. Author: Cosmin Lazar.
- inTrees** Interpret Tree Ensembles. Author: Houtao Deng.
- indicoio** A simple R Wrapper for the indico set of APIs. Author: Alexander Gedranovich. In view: [WebTechnologies](#).
- insuranceData** A Collection of Insurance Datasets Useful in Risk Classification in Non-life Insurance. Authors: Alicja Wolny-Dominiak and Michal Trzesiok.
- interAdapt** A shiny application for designing adaptive clinical trials Authors: Aaron Fisher, Michael Rosenblum, Harris Jaffee.
- interpretR** Binary Classifier Interpretation Functions. Authors: Michel Ballings and Dirk Van den Poel.
- ionflows** Calculate the Number of Required Flows for Semiconductor Sequencing. Authors: Michael Bockmayr and Jan Budczies.
- iosmooth** Functions for smoothing with infinite order flat-top kernels. Authors: Timothy L. McMurry, Dimitris N. Politis.
- ips** Interfaces to Phylogenetic Software in R. Author: Christoph Heibl.
- itertools2** Functions creating iterators for efficient looping. Authors: John A. Ramey, Kayla Schaefer.
- ivprobit** Instrumental variables probit model. Author: Zaghoudi Taha.
- jSonarR** jSonar Analytics Platform API for R. Author: jSonar Inc. In view: [WebTechnologies](#).
- jiebaR** Chinese Text Segmentation. Authors: Qin Wenfeng, and the authors of CppJieba for the included version of CppJieba.
- jomo** Multilevel Joint Modelling Multiple Imputation. Authors: Matteo Quartagno, James Carpenter.
- knockoff** Knockoff Filter. Authors: Rina Foygel Barber, Emmanuel Candes, Evan Patterson.

- kofnGA** A genetic algorithm for fixed-size subset selection. Author: Mark A. Wolters.
- kpodclustr** Method for Clustering Partially Observed Data. Authors: Jocelyn T. Chi [aut, cre], Eric C. Chi [ctb].
- kselection** Selection of k in k -means clustering. Author: Daniel Rodriguez Perez.
- lakemorpho** Lake morphometry in R. Author: Jeffrey W. Hollister [aut, cre].
- landpred** Landmark Prediction of a Survival Outcome. Author: Layla Parast.
- lazyeval** Lazy (Non-Standard) Evaluation. Authors: Hadley Wickham [aut, cre], RStudio [cph].
- lbfgs** Limited-memory BFGS Optimization. Authors: Antonio Coppola [aut, cre, cph], Brandon Stewart [aut, cph], Naoaki Okazaki [aut, cph], David Ardia [ctb, cph], Dirk Eddelbuettel [ctb, cph], Katharine Mullen [ctb, cph], Jorge Nocedal [ctb, cph]. In view: *Optimization*.
- leaderCluster** Leader Clustering Algorithm. Author: Taylor B. Arnold.
- lefse** Phylogenetic and Functional Analyses for Ecology. Author: Nathan G. Swenson.
- lintr** Static R Code Analysis. Author: Jim Hester [aut, cre].
- lm.beta** Add Standardized Regression Coefficients to lm-Objects. Author: Stefan Behrendt [aut, cre].
- lmeVarComp** Testing for a subset of variance components in linear mixed models. Author: Yichi Zhang.
- lmenssp** Linear Mixed Effects Models with Non-stationary Stochastic Processes. Authors: Ozgur Asar, Peter J. Diggle.
- lmmot** Multiple Ordinal Tobit (MOT) model. Author: Marvin N. Wright.
- logbin** Relative Risk Regression Using the Log Binomial Model. Author: Mark Donoghoe.
- logconPH** CoxPH Model with Log Concave Baseline Distribution. Author: Clifford Anderson-Bergman.
- logitchoice** Fitting L_2 -regularized logit choice models via generalized gradient descent. Author: Michael Lim.
- lpme** Local Polynomial Estimator in Measurement Error Models. Authors: Haiming Zhou and Xianzheng Huang.
- lucid** Lucid Printing of Floating Point Numbers. Author: Kevin Wright.
- lunar** Lunar Phase & Distance, Seasons and Other Environmental Factors. Author: Emmanuel Lazaridis [aut, cre].
- lvm4net** Latent Variable Models for Networks. Author: Isabella Gollini.
- m4fe** Models for Financial Economics. Author: Nathan Esau.
- maboost** Binary and Multiclass Boosting Algorithms. Author: Tofigh Naghibi.
- mads** Multi-Analysis Distance Sampling. Author: Laura Marshall.
- managelocalrepo** Manage a CRAN-style Local Repository. Author: Imanuel Costigan.
- manipulate** Interactive Plots for RStudio. Authors: JJ Allaire [aut, cre] (R interface), RStudio [cph].

- matR** Metagenomics Analysis Tools for R. Authors: Daniel Braithwaite [aut, cre], Kevin Keegan [aut], University of Chicago [cph].
- matchingMarkets** Analysis of stable matchings. Author: Thilo Klein.
- matpow** matrix powers. Authors: Norm Matloff, Jack Norman.
- mcc** Moment Corrected Correlation. Author: Yi-Hui Zhou.
- mdscore** Improved Score Tests for Generalized Linear Models. Authors: Antonio Hermes M. da Silva-Junior [aut, cre], Damiao N. da Silva [aut], Silvia L. P. Ferrari [ctb].
- mdsdt** Functions for Analysis of Data with General Recognition Theory. Authors: Robert X.D. Hawkins, Joe Houpt, Noah Silbert, Leslie Blaha, Thomas D. Wickens.
- measuRing** Detection and Control of Tree-Ring Widths on Scanned Image Sections of Dendrochronological Samples. Authors: Wilson Lara, Carlos Sierra.
- medflex** Flexible Mediation Analysis using Natural Effect Models. Authors: Johan Steen [aut, cre], Tom Loeyls [aut], Beatrijs Moerkerke [aut], Stijn Vansteelandt [aut], Joris Meys [ctb] (technical support), Theis Lange [ctb] (valuable suggestions).
- megaptera** MEGAPhylogeny Techniques in R. Author: Christoph Heibl.
- metaMix** Bayesian Mixture Analysis for Metagenomic Community Profiling. Author: Sofia Morfopoulou.
- metafolio** Metapopulation simulations for conserving salmon through portfolio optimization. Authors: Sean C. Anderson [aut, cre], Jonathan W. Moore [ctb], Michelle M. McClure [ctb], Nicholas K. Dulvy [ctb], Andrew B. Cooper [ctb].
- metaplus** Robust meta-analysis and meta-regression. Author: Ken Beath. In view: *Meta-Analysis*.
- meteoForecast** Numerical Weather Predictions. Authors: Oscar Perpignan Lamigueiro [cre, aut], Marcelo Pinho Almeida [ctb]. In view: *WebTechnologies*.
- mfblock** Fitting and Simulation of a block multifactor model. Author: Marc Fortier.
- micropan** Microbial Pan-genome Analysis. Authors: Lars Snipen and Kristian Hovde Liland.
- miniCRAN** Tools to create an internally consistent, mini version of CRAN with selected packages only. Authors: Revolution Analytics [aut], Andrie de Vries [aut, cre].
- mipfp** Multidimensional Iterative Proportional Fitting. Authors: Johan Barthelemy [aut, cre], Thomas Suesse [aut], Mohammad Namazi-Rad [ctb].
- mirtCAT** Computerized Adaptive Testing with Multidimensional Item Response Theory. Author: Phil Chalmers [aut, cre, cph]. In view: *Psychometrics*.
- mkde** 2D and 3D movement-based kernel density estimates (MKDEs). Authors: Jeff A. Tracey, James Sheppard, Jun Zhu, Robert Sinkovts, Amit Chourasia, Glenn Lockwood, and Robert N. Fisher. In view: *SpatioTemporal*.
- mnormpow** Multivariate Normal Distributions with Power Integrand. Authors: Alan Genz [ctb], Adelchi Azzalini [ctb], Alexis Bienvenüe [aut, cre], Christian Robert [aut].
- moonBook** Functions and Datasets for the Book by Keon-Woong Moon. Author: Keon-Woong Moon [aut, cre].
- mosaicData** Project MOSAIC (mosaic-web.org) data sets. Authors: Randall Pruim, Daniel Kaplan, Nicholas Horton.

- mp** Multidimensional Projection Techniques. Authors: Francisco M. Fatore, Samuel G. Fadel.
- mpath** Regularized Linear Models. Authors: Zhu Wang, with contributions from Achim Zeileis, Simon Jackman, Brian Ripley, Trevor Hastie, Rob Tibshirani, Balasubramanian Narasimhan, Gil Chu and Patrick Breheny.
- mpcv** Multivariate Process Capability Vector. Author: Krzysztof Ciupke.
- mppa** Statistics for analysing multiple simultaneous point processes on the real line. Authors: Patrick Rubin-Delanchy and Nicholas A Heard.
- msBP** Multiscale Bernstein Polynomials for Densities. Author: Antonio Canale.
- msda** Multi-class Sparse Discriminant Analysis. Authors: Qing Mai, Yi Yang, Hui Zou.
- mtk** Mexico ToolKit library (MTK). Authors: Juhui Wang [aut, cre] (software and engineering), Hervé Monod [aut] (applications and statistical methods), Robert Faivre [ctb] (applications and statistical methods), Hervé Richard [ctb] (software and engineering).
- multiband** Period Estimation for Multiple Bands. Authors: Eric C. Chi, James P. Long.
- multibiplotGUI** Multibiplot Analysis in R. Authors: Ana Belen Nieto Librero, Nora Baccala, Purificacion Vicente Galindo, Purificacion Galindo Villardon.
- multipleNCC** Weighted Cox-regression for nested case-control data. Authors: Nathalie C. Stoer, Sven Ove Samuelsen. In view: *Survival*.
- multispatialCCM** Multispatial Convergent Cross Mapping. Author: Adam Clark.
- multiwayvcov** Multi-way Standard Error Clustering. Authors: Nathaniel Graham and Mahmood Arai and Björn Hagströmer.
- musicNMR** Conversion of Nuclear Magnetic Resonance spectrum in audio file. Authors: Stefano Cacciatore, Edoardo Saccenti, Mario Piccioli.
- mvglmRank** Multivariate Generalized Linear Mixed Models for Ranking Sports Teams. Authors: Andrew T. Karl, Jennifer Broatch.
- mvprpb** Orthant Probability of the Multivariate Normal Distribution. Author: Noboru Nomura.
- mvrtn** Mean and Variance of Truncated Normal Distribution. Author: Matthew McLeod.
- mwaved** Multichannel wavelet deconvolution with additive long memory noise. Author: Justin Rory Wishart.
- myTAI** A Framework to Perform Phylotranscriptomics Analyses for Evolutionary Developmental Biology Research. Author: Hajk-Georg Drost.
- mycor** Automatic Correlation and Regression Test in a Data Frame. Author: Keon-Woong Moon [aut, cre].
- nabor** Wraps libnabo, a fast K Nearest Neighbour library for low dimensions. Authors: Stephane Mangenat (for libnabo), Gregory Jefferis.
- nasaweather** Collection of datasets from the ASA 2006 data expo. Author: Hadley Wickham [aut, cre].
- nat.nblast** NeuroAnatomy Toolbox (**nat**) Extension for Assessing Neuron Similarity and Clustering. Authors: Greg Jefferis and James Manton.
- nat.templatebrains** NeuroAnatomy Toolbox (**nat**) Extension for Handling Template Brains. Authors: James Manton and Greg Jefferis.

- netgsa** Network-based Gene Set Analysis. Authors: Ali Shojaie and Jing Ma.
- nettools** A Network Comparison Framework. Authors: Michele Filosi [aut, cre], Roberto Visintainer [aut], Samantha Riccadonna [aut], Giuseppe Jurman [ctb], Cesare Furlanello [ctb].
- networkD3** Tools for Creating D3 JavaScript Network Graphs from R. Authors: Christopher Gandrud, J.J. Allaire, and B.W. Lewis.
- nicheROVER** (Niche) (R)egion and Niche (Over)lap Metrics for Multidimensional Ecological Niches. Authors: Martin Lysy [aut, cre], Ashley D. Stasko [aut, ctb], Heidi K. Swanson [aut, ctb].
- nlWaldTest** Wald Test of nonlinear restrictions for regression parameters. Author: Oleh Komashko.
- nlsem** Fitting Structural Equation Mixture Models. Authors: Nora Umbach [aut, cre], Katharina Naumann [aut], David Hoppe [aut], Holger Brandt [aut], Augustin Kelava [ctb], Bernhard Schmitz [ctb].
- nodiv** Compares the Distribution of Sister Clades Through a Phylogeny. Author: Michael Krabbe Borregaard.
- nonnest2** Tests of Non-nested Models. Authors: Edgar C. Merkle and Dongjun You. In view: *Econometrics*.
- nontargetData** Quantized simulation data of isotope pattern centroids. Authors: Martin Loos, Francesco Corona.
- npsm** Nonparametric Statistical Methods using R. Authors: John Kloke, Joseph McKean.
- nscancer** Non-Negative and Sparse CCA. Authors: Christian Sigg [aut, cre], R Core team [aut].
- nsgp** Non-Stationary Gaussian Process Regression. Author: Markus Heinonen.
- nycflights13** Data about flights departing NYC in 2013. Author: Hadley Wickham [aut, cre].
- oaxaca** Blinder-Oaxaca decomposition. Author: Marek Hlavac.
- occ** Estimates PET neuroreceptor occupancies. Author: Joaquim Radua. In view: *Medical-Imaging*.
- onlinePCA** Online Principal Component Analysis. Author: David Degras.
- openssl** Bindings to OpenSSL. Author: Jeroen Ooms.
- opentraj** Tools for Creating and Analysing Air Trajectory Data. Author: Thalles Santos Silva.
- optiscale** Optimal scaling. Author: William G. Jacoby.
- optrees** Optimal Trees in Weighted Graphs. Author: Manuel Fontenla [aut, cre].
- orderedLasso** Ordered Lasso and Time-lag Sparse Regression. Authors: Jerome Friedman, Xiaotong Suo and Robert Tibshirani. In view: *TimeSeries*.
- ordinalgmifs** Ordinal Regression for High-dimensional Data. Authors: Kellie J. Archer, Jiayi Hou, Qing Zhou, Kyle Ferber, John G. Layne, Amanda Gentry.
- ore** An R Interface to the Oniguruma Regular Expression Library. Authors: Jon Clayden, based on Onigmo by K. Kosako and K. Takata.
- orgR** Analyse Text Files Created by Emacs' Org mode. Author: Yi Tang.

- pBrackets** Plot Brackets. Author: Andreas Schulz.
- pRSR** Test of periodicity using response surface regression. Author: M. S. Islam.
- packrat** A Dependency Management System for Projects and their R Package Dependencies. Authors: Kevin Ushey, Jonathan McPherson, Joe Cheng, JJ Allaire.
- panelaggregation** Aggregate Longitudinal Survey Data. Authors: Matthias Bannert [aut, cre], Gabriel Bucur [aut].
- parsedate** Recognize and Parse Dates in Various Formats, Including All ISO 8601 Formats. Authors: Gabor Csardi, Linus Torvalds.
- pauwels2014** Bayesian Experimental Design for Systems Biology. Author: Edouard Pauwels.
- pdR** threshold regression and unit root test in panel data. Author: Ho Tsung-wu.
- pedometrics** Pedometric Tools and Techniques. Authors: Alessandro Samuel-Rosa [aut, cre], Lucia Helena Cunha dos Anjos [ths], Gustavo de Mattos Vasques [ths], Gerard B. M. Heuvelink [ths], Tony Olsen [ctb], Tom Kincaid [ctb], Juan Carlos Ruiz Cuetos [ctb], Maria Eugenia Polo Garcia [ctb], Pablo Garcia Rodriguez [ctb], Edzer Pebesma [ctb], Jon Skoien [ctb], Joshua French [ctb].
- penMSM** Estimating regularized multistate models using L_1 penalties. Author: Holger Reulen. In view: *Survival*.
- pez** Phylogenetics for the Environmental Sciences. Authors: William D. Pearse, Marc W. Cadotte, Jeannine Cavender-Bares, Caroline Tucker, Steve C. Walker, Matthew R. Helmus.
- phylin** Spatial interpolation of genetic data. Authors: Pedro Tarroso, Guillermo Velo-Anton, Silvia Carvalho.
- picasso** Pathwise Calibrated Sparse Shooting Algorithm. Authors: Xingguo Li, Tuo Zhao and Han Liu.
- pingr** Check If a Remote Computer is Up. Author: Gabor Csardi [aut, cre].
- plotROC** Generate Useful ROC Curve Charts for Print and Interactive Use. Author: Michael C Sachs.
- pointdensityP** Point density for geospatial data. Authors: Paul Evangelista and Dave Beskow.
- polidata** Political Data Interface in R. Authors: Eunjeong Park [aut, cre], Jong Hee Park [aut]. In view: *WebTechnologies*.
- popKorn** For interval estimation of mean of selected populations. Authors: Vik Gopal, Claudio Fuentes.
- poplite** Tools for simplifying the population and querying of SQLite databases. Author: Daniel Bottomly.
- powerr** Analysis of Power System. Author: Jingfan Sun [aut, cre].
- prc** Paired Response Curve. Author: Youyi Fong.
- predictmeans** Calculate Predicted Means for Linear Models. Authors: Dongwen Luo, Siva Ganesh and John Koolaard.
- preseqR** Predicting Species Accumulation. Authors: Chao Deng, Timothy Daley and Andrew D. Smith.
- prettyunits** Pretty, Human Readable Formatting of Quantities. Author: Gabor Csardi [aut, cre].

- propOverlap** Feature (gene) selection based on the Proportional Overlapping Scores. Authors: Osama Mahmoud, Andrew Harrison, Aris Perperoglou, Asma Gul, Zardad Khan, Berthold Lausen.
- proteomics** Statistical Analysis of High Throughput Proteomics Data. Author: Thomas W. D. Möbius.
- pryr** Tools for computing on the language. Authors: Hadley Wickham [aut, cre], R Core Team [ctb].
- pxweb** R interface to the PX-Web/PC-Axis API. Authors: Mans Magnusson, Leo Lahti, Love Hansson. In view: *WebTechnologies*.
- qdapRegex** Regular Expression Removal, Extraction, and Replacement Tools. Authors: Jason Gray [ctb], Tyler Rinker [aut, cre].
- qdm** Fitting a Quadrilateral Dissimilarity Model to Same-Different Judgments. Authors: Nora Umbach [aut, cre], Florian Wickelmaier [aut].
- qgtools** Tools for Quantitative Genetics Data Analyses. Authors: Jixiang Wu, Johnie N. Jenkins and Jack C. McCarty.
- qicharts** Quality Improvement Charts. Authors: Jacob Anhoej [aut, cre], Timo Roeder [ctb].
- qqtest** Quantile Quantile Plots Self Calibrating For Visual Testing. Author: Wayne Oldford [aut, cre].
- quad** Exact permutation moments of quadratic form statistics. Author: Yi-Hui Zhou.
- qualCI** Causal Inference with Qualitative and Ordinal Information on Outcomes. Authors: Konstantin Kashin, Adam Glynn, Nahomi Ichino.
- rARPACK** R wrapper of ARPACK for large scale eigenvalue/vector problems, on both dense and sparse matrices. Authors: Yixuan Qiu, Jiali Mei and authors of the ARPACK library. In view: *NumericalMathematics*.
- rDEA** Robust Data Envelopment Analysis (DEA) for R. Authors: Jaak Simm [aut, cre], Galina Besstremyannaya [aut].
- rFDSN** Get Seismic Data from the International Federation of Digital Seismograph Networks. Author: Daniel C. Bowman [aut, cre]. In view: *WebTechnologies*.
- rJPSGCS** R-interface to Gene Drop Simulation from JPSGCS. Authors: Sigal Blay [aut], Jinko Graham [aut], Brad McNeney [aut, cre], Annick Nembot-Simo [aut], Alun Thomas [ctb], Hin-Tak Leung [ctb].
- rPref** Routines to select and visualize the maxima for a given strict partial order. Author: Patrick Rocks.
- rSPACE** Spatially-Explicit Power Analysis for Conservation and Ecology. Authors: Martha Ellis, Jake Ivan, Jody Tucker, Mike Schwartz.
- rYotheria** Access to the YouTheria mammal trait database. Author: Tom August. In view: *WebTechnologies*.
- radar** Fundamental Formulas for Radar. Authors: Jose' Gama [aut, cre], Nick Guy [aut].
- rappdirs** Application directories: determine where to save data, caches and logs. Authors: Hadley Wickham [trl, cre, cph], RStudio [cph], Sridhar Ratnakumar [aut], Trent Mick [aut], ActiveState [cph], Eddy Petrisor [ctb], Trevor Davis [trl, aut], Gabor Csardi [ctb], Gregory Jefferis [ctb].
- rareGE** Testing Gene-Environment Interaction for Rare Genetic Variants. Author: Han Chen.

- rcbalance** Large, Sparse Optimal Matching with Refined Covariate Balance. Author: Samuel D. Pimentel.
- rcicr** Reverse correlation image classification toolbox. Author: Ron Dotsch.
- rclinicaltrials** Download Aggregate Trial Information and Results from ClinicalTrials.gov. Author: Michael C Sachs. In view: *WebTechnologies*.
- rcrossref** R Client for Various CrossRef APIs. Authors: Carl Boettiger [aut], Ted Hart [aut], Scott Chamberlain [aut, cre], Karthik Ram [aut].
- reconstructr** Session Reconstruction and Analysis. Author: Oliver Keyes.
- recosystem** Recommender System using Matrix Factorization. Authors: Yixuan Qiu, Chih-Jen Lin, Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin and other contributors.
- redcapAPI** R interface to REDCap. Authors: Benjamin Nutter. Initiated by Jeffrey Horner and Will Gray with contributions from Jeremy Stephens, and Will Beasley. In view: *WebTechnologies*.
- refset** Subsets with Reference Semantics. Author: David Hugh-Jones.
- refund.wave** Wavelet-Domain Regression with Functional Data. Authors: Lan Huo, Philip Reiss and Yihong Zhao.
- regexr** Readable Regular Expressions. Author: Tyler Rinker [aut, cre].
- remote** Empirical Orthogonal Teleconnections in R. Authors: Tim Appelhans, Florian Detsch, Thomas Nauss.
- replicationInterval** Replication Interval Functions. Author: David Stanley.
- repra** Renewable Energy Probability Resource Assessment Tool (REPRA). Authors: Eduardo Ibanez [aut, cre], National Renewable Energy Laboratory [cph].
- retistruct** Retinal Reconstruction Program. Authors: David C. Sterratt [aut, cre, cph], Daniel Lyngholm [aut, cph].
- rex** Friendly Regular Expressions. Authors: Kevin Ushey [aut], Jim Hester [cre, aut].
- rfUtilities** Random Forests model selection and performance evaluation. Authors: Jeffrey S. Evans [aut, cre], Melanie A. Murphy [aut].
- rfoaas** R Interface to FOAAS. Author: Dirk Eddelbuettel.
- rfordummies** Code examples to accompany the book “R for Dummies”. Authors: Andrie de Vries [aut, cre], Joris Meys [aut].
- riskSimul** Risk Quantification for Stock Portfolios under the *t*-Copula Model. Authors: Wolfgang Hormann, Ismail Basoglu. In view: *Finance*.
- rivernet** Read, Analyze and Plot River Networks. Author: Peter Reichert.
- rjstat** Read and write JSON-stat data sets. Authors: Aaron Schumacher, Håkon Malmedal.
- rkvo** Read Key/Value Pair Observations. Author: Vehbi Sinan Tunalioglu [aut, cre].
- rmarkdown** Dynamic Documents for R. Authors: JJ Allaire, Jonathan McPherson, Yihui Xie, Hadley Wickham, Joe Cheng, Jeff Allen.
- rnaseqWrapper** Wrapper for several R packages and scripts to automate RNA-seq analysis. Author: Mark Peterson.
- rnbm** Access NBN Data. Authors: Stuart Ball and Tom August. In view: *WebTechnologies*.

- rncl** An interface to the Nexus Class Library. Authors: Francois Michonneau [aut, cre], Ben Bolker [aut], Mark Holder [aut, cre], Paul Lewis [aut, cre], Brian O'Meara [aut, cre].
- rnrfa** UK National River Flow Archive data from R. Authors: Claudia Vitolo, Matthew Fry. In view: *WebTechnologies*.
- robustDA** Robust Mixture Discriminant Analysis. Authors: Charles Bouveyron and Stephane Girard. In view: *Robust*.
- robustvarComp** Robust Estimation of Variance Component Models. Authors: Claudio Agostinelli and Victor J. Yohai.
- rodd** Optimal Discriminating Designs. Author: Roman Guchenko [aut, cre].
- ropensecretsapi** Interface for the OpenSecrets.org API. Author: Thomas P. Fuller. In view: *WebTechnologies*.
- rowr** Row-based functions for R objects. Author: Craig Varrichio.
- rpg** Easy interface to advanced PostgreSQL features. Author: Timothy H. Keitt.
- rplexos** Read and Analyze PLEXOS Solutions from R. Authors: Eduardo Ibanez [aut, cre], National Renewable Energy Laboratory [cph].
- rqPen** Penalized Quantile Regression. Author: Ben Sherwood.
- rsdmx** Tools for Reading SDMX Data and Metadata. Authors: Emmanuel Blondel [aut, cre], Matthieu Stigler [ctb]. In view: *WebTechnologies*.
- rsml** Plant Root System Markup Language (RSML) File Processing. Author: Guillaume Lobet.
- rstackdeque** Persistent Fast Amortized Stack and Queue Data Structures. Author: Shawn T. O'Neil.
- rsubgroup** Subgroup Discovery and Analytics. Author: Martin Atzmueller.
- rsunlight** Interface to Sunlight Foundation APIs. Authors: Scott Chamberlain [aut, cre], Thomas J. Leeper [ctb]. In view: *WebTechnologies*.
- rtdists** Response time distributions. Authors: Scott Brown [aut], Matthew Gretton [aut], Andrew Heathcote [aut], Andreas Voss [ctb], Jochen Voss [ctb], Andrew Terry [ctb], Henrik Singmann [aut, cre].
- rtkpp** STK++ Integration To R Using **Rcpp**. Authors: Serge Iovleff [aut, cre], Vincent Kubicki [ctb], Quentin Grimonprez [ctb], Parmeet Bhatia [ctb].
- rtype** A strong type system for R. Author: Kun Ren.
- rucm** Implementation of Unobserved Components Model (UCM) in R. Author: Kaushik Roy Chowdhury.
- ruv** Detect and Remove Unwanted Variation using Negative Controls. Author: Johann Gagnon-Bartsch.
- rvalues** Computation of r-values for ranking in high-dimensional settings. Authors: Nicholas Henderson and Michael Newton.
- rvest** Easily Harvest (Scrape) Web Pages. Authors: Hadley Wickham [aut, cre], RStudio [cph].
- saccades** Detection of Fixations in Eye-Tracking Data. Author: Titus von der Malsburg [aut, cph, cre].

- sads** Maximum Likelihood Models for Species Abundance Distributions. Authors: Paulo I. Prado and Murilo Dantas Miranda.
- saeSim** Simulation Tools for Small Area Estimation. Author: Sebastian Warnholz.
- saery** Small Area Estimation for Rao and Yu Model. Authors: Maria Dolores Esteban Lefler, Domingo Morales Gonzalez, Agustin Perez Martin.
- safi** Sensitivity Analysis for Functional Input. Authors: Jana Fruth, Malte Jastrow.
- sanitizers** C/C++ source code to trigger Address and Undefined Behaviour Sanitizers. Author: Dirk Eddelbuettel.
- saps** Significance Analysis of Prognostic Signatures. Authors: Daniel Schmolze [aut, cre], Andrew Beck [aut], Benjamin Haibe-Kains [aut].
- scrm** Simulating the Evolution of Biological Sequences. Authors: Paul Staab [aut, cre, cph], Zhu Sha [aut, cph], Dirk Metzler [aut, cph, ths], Gerton Lunter [aut, cph, ths].
- sdCTarget** Statistical Disclosure Control Substitution Matrix Calculator. Author: Emmanuel Lazaridis [aut, cre].
- seclinear** Spatially Explicit Capture-Recapture for Linear Habitats. Author: Murray Efford.
- seedy** Simulation of Evolutionary and Epidemiological Dynamics. Author: Colin Worby.
- segmag** Determine Event Boundaries in Event Segmentation Experiments. Authors: Frank Papenmeier [aut, cre], Konstantin Sering [ctb].
- selectspm** Select point patterns models based on minimum contrast and AIC. Author: Marcelino de la Cruz.
- selfingTree** Genotype Probabilities in Intermediate Generations of Inbreeding Through Selfing. Authors: Frank Technow [aut, cre].
- sensory** Simultaneous Model-Based Clustering and Imputation via a Progressive Expectation-Maximization (PEM) algorithm. Authors: Brian C. Franczak, Ryan P. Browne and Paul D. McNicholas.
- settings** Software Option Settings Manager for R. Author: Mark van der Loo.
- shopifyr** An R Interface to the Shopify API. Author: Charlie Friedemann. In view: *WebTechnologies*.
- sievetest** Sieve test reporting functions. Author: Petr Matousu.
- sigloc** Signal Location Estimation. Author: Sergey S. Berg.
- signal.hsmm** Predict Presence of Signal Peptides. Authors: Michal Burdukiewicz [cre, aut], Piotr Sobczyk [aut].
- simPop** Simulation of Synthetic Populations for Survey Data Considering Auxiliary Information. Authors: Bernhard Meindl, Matthias Templ, Andreas Alfons, Alexander Kowarik, with contributions from Mathieu Ribatet.
- simTool** Conduct Simulation Studies with a Minimal Amount of Source Code. Author: Marsel Scheer.
- simrel** Linear Model Data Simulation and Design of Computer Experiments. Author: Solve Sæbø.
- sisVIVE** Some Invalid Some Valid Instrumental Variables Estimator. Author: Hyunseung Kang.

- sla** Two-Group Straight Line ANCOVA. Authors: W. Gregory Alvord [aut, cre], Nick Carchedi [aut].
- slackr** Send messages, images, R objects and files to Slack.com channels/users. Authors: Bob Rudis and Jay Jacobs. In view: *WebTechnologies*.
- sld** Estimation and Use of the Quantile-Based Skew Logistic Distribution. Authors: Robert King, Paul van Staden.
- slp** Discrete Prolate Spheroidal (Slepian) Sequence Regression Smoothers. Authors: Wesley Burr, with contributions from Karim Rahim.
- smac** Sparse Multi-category Angle-Based Large-Margin Classifiers. Authors: Chong Zhang, Guo Xian Yau, Yufeng Liu.
- smacpod** Statistical Methods for the Analysis of Case-Control Point Data. Author: Joshua French.
- smnet** Smoothing For Stream Network Data. Author: Alastair Rushworth.
- smoother** Gaussian window smoothing. Author: Nicholas Hamilton.
- snht** Standard Normal Homogeneity Test. Author: Josh Browning.
- snipEM** Snipping methods for robust estimation and clustering. Authors: Alessio Farcomeni, Andy Leung.
- snpRF** Random Forest for SNPs to Prevent X-chromosome SNP Importance Bias. Authors: Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener. Modifications of randomForest v 4.6-7 for SNPs by Greg Jenkins based on a method developed by Stacey Winham, Greg Jenkins and Joanna Biernacka.
- snpur** Supplementary Non-parametric Statistics Methods. Author: Debin Qiu.
- sns** Stochastic Newton Sampler. Authors: Alireza S. Mahani, Asad Hasan, Marshall Jiang, Mansour T.A. Sharabiani.
- sonicLength** Estimating Abundance of Clones from DNA fragmentation data. Author: Charles Berry.
- spThin** Functions for Spatial Thinning of Species Occurrence Records for Use in Ecological Models. Authors: Matthew E. Aiello-Lammens, Robert A. Boria, Aleksandar Radosavljevic, Bruno Vilela, and Robert P. Anderson.
- spanr** Search Partition Analysis. Author: Roger Marshall.
- spareserver** Client side load balancing. Author: Gabor Csardi [aut, cre].
- spark** Sparklines in the R Terminal. Author: Gabor Csardi [aut, cre].
- sparseSEM** Sparse-aware Maximum Likelihood for Structural Equation Models. Author: Anhui Huang.
- spatialnbda** Performs spatial NBDA in a Bayesian context. Author: Glenna Nightingale.
- spca** Sparse Principal Component Analysis. Author: Giovanni Maria Merola.
- spdynmod** Spatio-dynamic wetland plant communities model. Authors: Javier Martinez-Lopez, Babak Naimi, Julia Martinez-Fernandez.
- spiders** Fits predator preferences model. Authors: Edward A. Roualdes and Simon Bonner.
- spnet** Plotting (social) networks on maps. Authors: Emmanuel Rousseaux, Marion Deville.
- sprinter** Framework for Screening Prognostic Interactions. Author: Isabell Hoffmann.

- sprm** Sparse and Non-Sparse Partial Robust M Regression. Authors: Sven Serneels, BASF Corp and Irene Hoffmann.
- sptm** SemiParametric Transformation Model Methods. Authors: Youyi Fong, Krisztian Sebestyen.
- srd** Draws Scaled Rectangle Diagrams. Author: Roger Marshall.
- ssd** Sample Size Determination (SSD) for Unordered Categorical Data. Authors: Junheng Ma and Jiayang Sun.
- ssh.utils** Local and remote system commands with output and error capture. Author: Sergei Izrailev.
- stabs** Stability Selection with Error Control. Authors: Benjamin Hofner [aut, cre], Torsten Hothorn [aut]. In view: *MachineLearning*.
- statar** Tools Inspired by Stata to Clean, Explore and Join Datasets. Author: Matthieu Gomez [aut, cre].
- statcheck** Extract statistics from articles and recompute p -values. Authors: Sacha Epskamp and Michele B. Nuijten.
- statebins** An alternative to choropleth maps for USA States. Author: Bob Rudis.
- strataG** Summaries and Population Structure Analyses of Haplotypic and Genotypic Data. Author: Eric Archer.
- subgroup** Methods for exploring treatment effect heterogeneity in subgroup analysis of clinical trials. Author: I. Manjula Schou.
- subrank** Computes copula using ranks and subsampling. Author: Jerome Collet.
- sudokuAlt** Tools for Making, Displaying and Spoiling Sudoku Games. Author: Bill Venables.
- summarytools** Dataframe Summaries, Frequency Tables and Numerical Summaries with Customizable Output. Author: Dominic Comtois.
- survAccuracyMeasures** Estimate accuracy measures for risk prediction markers from survival data. Authors: Yingye Zheng, Tianxi Cai, and Marshall Brown. In view: *Survival*.
- svgViewR** 3D Animated Interactive Visualizations using SVG. Author: Aaron Olsen.
- sweidnumbr** Handling of Swedish Identity Numbers. Author: Mans Magnusson.
- swfscMisc** Miscellaneous Functions for Southwest Fisheries Science Center. Author: Eric Archer.
- synthpop** Generating synthetic versions of sensitive microdata for statistical disclosure control. Authors: Beata Nowok, Gillian M Raab and Chris Dibben.
- tcR** Data Analysis of T-cell Receptor Repertoires. Authors: Vadim Nazarov, Mikhail Pogorelyy.
- testthatsomemore** A mocking, stubbing, and file testing framework extending **testthat**. Author: Robert Krzyzanowski [aut, cre].
- textreg** n-gram Text Regression, aka Concise Comparative Summarization. Author: Luke Miratrix.
- thermocouple** Temperature Measurement with Thermocouples, RTD and IC Sensors. Author: Jose Gama [aut, cre].

- tidyr** Easily Tidy Data with spread() and gather() Functions. Authors: Hadley Wickham [aut, cre], RStudio [cph].
- timeseriesdb** Store and Organize Time Series in a Database. Author: Matthias Bannert [aut, cre].
- timma** Target Inhibition Interaction using Maximization and Minimization Averaging. Author: Liye.
- tlm** Effects under Linear, Logistic and Poisson Regression Models with Transformed Variables. Authors: Jose Barrera-Gomez and Xavier Basagana.
- tmap** Thematic Maps. Author: Martijn Tennekes. In view: *OfficialStatistics*.
- toxtestD** Experimental design for binary toxicity tests. Authors: Nadia Keddig and Werner Wosniok.
- traj** Trajectory Analysis. Authors: Marie-Pierre Sylvestre, Dan Vatnik.
- trajectories** Classes and methods for trajectory data. Authors: Edzer Pebesma [cre, aut], Benedikt Klus [aut].
- translateR** Bindings for the Google and Microsoft Translation APIs. Authors: Christopher Lucas and Dustin Tingley. In view: *WebTechnologies*.
- treatSens** Sensitivity analysis for causal inference. Authors: Nicole Bohme Carnegie, Masataka Harada, Jennifer Hill.
- treeclim** Numerical calibration of proxy-climate relationships. Authors: Christian Zang [aut, cre, cph, trl], Franco Biondi [ctb, cph].
- trimTrees** Trimmed opinion pools of trees in a random forest. Authors: Yael Grushka-Cockayne, Victor Richmond R. Jose, Kenneth C. Lichtendahl Jr. and Huanghui Zeng, based on the source code from the **randomForest** package by Andy Liaw and Matthew Wiener and on the original Fortran code by Leo Breiman and Adele Cutler.
- trotter** Pseudo-Vectors Containing All Permutations, Combinations and Subsets of Objects Taken from a Vector. Author: Richard Ambler.
- tsc** Likelihood-ratio Tests for Two-Sample Comparisons. Authors: Yang Zhao, Albert Vexler, Alan Hutson.
- tuftehandout** Tufte-style HTML document format for rmarkdown. Author: Michael Sachs.
- tumblrR** Tumblr API. Author: Andrea Capozio. In view: *WebTechnologies*.
- tuple** Find every match, or orphan, duplicate, triplicate, or other replicated values. Author: Emmanuel Lazaridis [aut, cre].
- tvd** Total Variation Denoising. Author: Mark Pinese [aut, cre, cph].
- uniftest** Tests for Uniformity. Authors: Maxim Melnik, Ruslan Pusev.
- unittest** TAP-compliant unit testing. Authors: Jamie Lentin [aut, cre], Anthony Hennessey [aut].
- urltools** Vectorised Tools for URL Handling and Parsing. Author: Oliver Keyes.
- uskewFactors** Model-based clustering via mixtures of unrestricted skew-*t* factor analyzer models. Authors: Paula M. Murray, Ryan P. Browne, and Paul D. McNicholas.
- vartors** Transform Definition of Variables to R Scripts. Authors: Joris Muller [aut, cre], Erik-André Sauleau [ctb].

- vcrpart** Tree-Based Varying Coefficient Regression for Generalized Linear and Ordinal Mixed Models. Authors: Reto Buergin [aut, cre, cph], Gilbert Ritschard [ctb, ths]. In view: *MachineLearning*.
- vdmR** Visual Data Mining Tools for R. Author: Tomokazu Fujino.
- vegan3d** Static and dynamic 3D plots for **vegan** package. Authors: Jari Oksanen [aut, cre], Roeland Kindt [aut], Gavin L. Simpson [aut].
- vetools** Tools for Venezuelan Environmental Data. Authors: Andrew Sajo-Castelli [aut, cre], Desiree Villalta [ctb], Lelys Bravo [ctb].
- virtualespecies** Generation of Virtual Species Distributions. Authors: Boris Leroy with help from C. N. Meynard, C. Bellard and F. Courchamp.
- visova** Visualization of Analysis of Variance. Author: Ali Vala Barbaros.
- vudc** Visualization of Univariate Data for Comparison. Author: Csaba Farago.
- webutils** Utility Functions for Web Applications. Author: Jeroen Ooms. In view: *WebTechnologies*.
- wikipediatrend** Public Subject Attention via Wikipedia Page Access Statistics. Authors: Peter Meissner [aut, cre], R Core team [ctb]. In view: *WebTechnologies*.
- wildlifeDI** Calculate Indices of Dynamic Interaction for Wildlife Telemetry Data. Author: Jed Long. In view: *SpatioTemporal*.
- windex** Analysing convergent evolution using the Wheatsheaf index. Authors: Kevin Arbuckle and Amanda Minter.
- wrassp** Interface to the ASSP Library. Authors: Raphael Winkelmann [aut, cre], Lasse Bombien [aut], Michel Scheffers [aut].
- wux** Wegener Center Climate Uncertainty Explorer. Authors: Thomas Mendlik, Georg Heinrich, Andreas Gobiet and Armin Leuprecht.
- x.ent** eXtraction of ENTity. Authors: Nicolas Turenne, Tien T. Phan.
- xgboost** eXtreme Gradient Boosting. Authors: Tianqi Chen, Tong He.
- xtal** Crystallization Toolset. Authors: Qingan Sun, Xiaojun Li.
- ztable** Zebra-Striped Tables in L^AT_EX and HTML Formats. Author: Keon-Woong Moon [aut, cre].

Other changes

The following packages were moved to the Archive: ARAMIS, ARTIVA, AWS.tools, Ace, AncestryMapper, Animal, BLCOP, BMamevt, BOG, BaSAR, Barnard, BayesPen, BayesQTLBIC, Biomarker, CCMnet, CFL, CONOR, CONORData, CVD, DRI, DWD, Defaults, Devore6, DierckxSpline, ENA, ESPRESSO, EVER, FPDC, FRBData, Fused-Lasso, GOSummaries, HIBAG, HPO.db, HPOSim, Jjcorr, KsPlot, LargeRegression, MASSI, MCMChybridGP, MFSAS, Mifuns, MLEP, MLPastats, MMST, MVPARTwrap, MVpower, Mangrove, MiClip, MiscPsycho, MortalitySmooth, NHMMfdr, NMRS, Nemenyi, NetPreProc, OneHandClapping, PIN, PKmodelFinder, PKtools, POET, PSMix, PamGeneMixed, RAHRS, RBerkeley, RC, RDS, RECISO, REGENT, RHmm, RJSONLD, RMendeley, RMessenger, RNCBI, RNCBIAxis2Libs, RPPanalyzer, RSQLite.extfuns, RSocrata, RSpincalc, RcmdrPlugin.doBy, Rearrangement, RelativeRisk, Reot, Rgbp, Rgnuplot, Rjms, Rjmsjars, Rknots, SCEPtERextras, SDBP, SDDA, SRPM, STARSEQ, SesIndexCreator, SigTree, SimHap, StVAR, StateTrace, StreamingLm, TFDEA, TRIANG, TRIANGG, TSAgg, UBCRM, VisuClust, WebDevelopR, aCGH.Spline, acer,

anaglyph, anoint, ant, attfad, avgrankoverlap, biOps, biOpsGUI, bigRR, bigml, bilan, blockTools, bootfs, cacheSweave, cacher, capme, cgwtools, changeLOS, colbycol, comorbidities, complex.surv.dat.sim, confReg, conicfit, csound, cusp, dataone, dataonelib, ddepn, dse1, dse2, easi, edmr, em2, emg, emplik2, factas, faisalconjoint, fork, fracprolif, gWidgetsWWW, gammSlice, gcExplorer, genomicper, glmperm, googlePublicData, gmaRt, highriskzone, iScreen, icomp, ipw, iwtp, jmec, lancet.iraqmortality, latticist, ldDesign, lmPerm, lmbc, logistiX, logregperm, longclust, malaria.em, margie, marginalmodelplots, maticce, metrumrg, mixedQF, mixstock, modelcf, mrdrc, mseq, muRL, multicore, munsellinterpol, mvpart, mvtBinaryEP, nacopula, neuRosim, noia, nordklimdata1, nricens, oposSOM, optmatch, orthogonalsplinebasis, pacose, parspatstat, pcaPA, pccenum, pfa, pgnorm, phyext, playitbyr, plink, postCP, ppmlasso, ppstat, processdata, qfa, randomSurvivalForest, rcqp, rdyncall, review, rgexf, risaac, rknn, rmmseg4j, rocplus, roxygen, rrdf, rrdflibs, rwm, samplesize, scaleboot, seqminer, signalextraction, sigora, sltl, speedRlibTF, speedRlibs, sra, ssize.fdr, standGL, sugaR, survBayes, survJamda, survJamda.data, sweSCB, tilting, transmission, transnet, treelet, varcompci, vcf2geno, vimcom, visualizationTools, waldwolf, wccsom, weathermetrics, wild1, xgrid

The following packages were resurrected from the Archive: **BTSPAS**, **BayesTree**, **BcDiag**, **CGP**, **EstSimPDMP**, **HDPenReg**, **HiDimDA**, **LogicForest**, **MAVTgsa**, **OrdLogReg**, **PHYLOGR**, **RcmdrPlugin.FactoMineR**, **RcmdrPlugin.steepness**, **RecordLinkage**, **SparseTSCGM**, **SpatialPack**, **XLConnectJars**, **agsemisc**, **convevol**, **ecp**, **evtree**, **pgmm**, **rngSetSeed**, **sensitivityPStrat**, **sprint**, **tagcloud**, **tclust**

The following packages had to be removed: **GWAtoolbox**, **Gmisc**, **LDEplorer**, **Rdrools**, **Rdroolsjars**, **WhopGenome**, **bstats**

Kurt Hornik

WU Wirtschaftsuniversität Wien, Austria

Kurt.Hornik@R-project.org

Achim Zeileis

Universität Innsbruck, Austria

Achim.Zeileis@R-project.org

Changes in R

In version 3.1.2

by the R Core Team

CHANGES IN R 3.1.2

NEW FEATURES

- `embedFonts()` now defaults to `format = "ps2write"` for `‘.ps’` and `‘.eps’` files. This is available in Ghostscript 9.x (since 2010) whereas the previous default, `format = "pswrite"`, was removed in Ghostscript 9.10.
- For consistency with `[dpqr]norm()`, `[dp]lnorm(sdlog = 0)` model a point mass at `exp(mu log)` rather than return `NaN` (for an error).
- `capabilities()` now reports if ICU is compiled in for use for collation (it is only actually used if a suitable locale is set for collation, and never for a C locale).
- (OS X only.) Package `tktk` checks when loaded if it is linked against the CRAN X11-based Tcl/Tk and if so that the Tcl/Tk component and the X11 libraries are installed. This allows more informative error messages to be given advising the installation of the missing component or of XQuartz.

The X11() device and X11-based versions of the data editor and viewer (invoked by `edit()` and `View()` for data frames and matrices from command-line R) check that the X11 libraries are installed and if not advises installing XQuartz.

- `icuSetCollate()` allows `locale = "default"`, and `locale = "none"` to use OS services rather than ICU for collation.
Environment variable `R_ICU_LOCALE` can be used to set the default ICU locale, in case the one derived from the OS locale is inappropriate (this is currently necessary on Windows).
- New function `icuGetCollate()` to report on the ICU collation locale in use (if any).
- `utils::urlencode()` was updated to use unreserved and reserved characters from RFC 3986, <http://tools.ietf.org/html/rfc3986>, instead of RFC 1738.
- `unique(warnings())` and `c(warnings())` are now supported.
- The Bioconductor `‘version’` used by `setRepositories()` now defaults to 3.0. (It can be set at runtime *via* environment variable `R_BIOC_VERSION`.)

INSTALLATION and INCLUDED SOFTWARE

- The configure script reports on the more important capabilities/options which will not be compiled in.
More types of external BLAS are recognized by name in that report.
- When building R as a shared library, the `‘-L${R_HOME}/lib${R_ARCH}’` flag is placed earlier in the link commands used during installation and when packages are installed: this helps ensure that the current build has priority if an R shared library has already been installed by e.g. `install-libR` in a library mentioned in `LD_FLAGS` (and not in `‘your system’s library directory’` as documented). (Wish of PR#15790.)
- LaTeX package `upquote` is no longer required for R’s use of `inconsolata`.
- (Windows only) If both 32 and 64 bit versions of R are installed, the `‘bin/R.exe’` and `‘bin/Rscript.exe’` executables now run 64 bit R. (To run 32 bit R, overwrite these files with copies of `‘bin/i386/Rfe.exe’`.)

UTILITIES

- Running R CMD check with `_R_CHECK_DEPENDS_ONLY_` true now makes the ‘VignetteBuilder’ packages available even if they are listed in ‘Suggests’, since they are needed to recognise and process non-Sweave vignettes.
- R CMD check now reports empty `importFrom` declarations in a ‘NAMESPACE’ file, as these are common errors (writing `importFrom(Pkg)` where `import(Pkg)` was intended).
- R CMD check now by default checks code usage directly on the package namespace without loading and attaching the package and its suggests and enhances. For good practice with packages in the ‘Suggests’ field, see §1.1.3.1 of ‘Writing R Extensions’. For use of lazy-data objects in the package’s own code, see `?data`.

BUG FIXES

- `dmultinom()` did not handle non-finite probabilities correctly.
- `prettyNum(x, zero.print=*)` now also works when `x` contains NAs.
- A longstanding bug exhibited by `nLminb()` on Windows was traced to a compiler bug in gcc 4.6.3; a workaround has been put in place. (PR#15244 and PR#15914).
- Rendering of `\command` in HTML versions of help pages has been improved: this is particularly evident on the help page for `INSTALL`.
- `as.hexmode(x)` and `as.octmode(x)` now behave correctly for some numeric `x`, e.g., `c(NA, 1)` or `c(1, pi)`.
- `drop1()` failed if the scope argument had no variables to drop. (PR#15935)
- `edit()` (and hence `fix()`) failed if an object had a non-character attribute named “source” (an attribute that had been used in R prior to version 2.14.0).
- `callGeneric()` could fail if the generic had `...` as a formal argument. (PR#15937).
- Forking in package **parallel** called C entry point `exit` in the child. This was unsafe (`_exit` should have been called), and could flush `stdin` of the main R process (seen most often on Solaris).
As good practice, `stdout` is now flushed before forking a child.
- R objects such as `list(`a\b` = 1)` now print correctly.
- `getAnywhere("C_pbinom")` now returns correctly a single object (rather than unlisting it).
- The `confint()` method for `nls()` fits failed if these has specified parameter limits despite using an algorithm other than “port”. (PR#15960)
- Subclassing an S4 class failed if the class required arguments to the generator, through its `initialize()` method.
- `removeSource()` did not properly handle expressions containing arguments that were supplied as missing, e.g. `x[i,]`. (PR#15957)
- `as.environment(list())` now works, and `as.list()` of such an environment is now the same as `list()`.
- Several `tcltk` functions failed when run in unusual environments. (PR#15970)
- `options(list())` now works (trivially). (PR#15979)

- `merge(<dendrogram>, . . .)` now works correctly for two ‘independent’ dendrograms (PR#15648), and still compatibly via `adjust = "auto"` e.g. for two branches of an existing dendrogram.
- The `plot` method for `"hclust"` objects gets an optional argument `check`; When that is `true` (the default) it checks more carefully for valid input.
- (Windows only) If a user chose to install 64 bit R but not 32 bit R, the ‘bin/R’ and ‘bin/Rscript’ executables failed to run. (PR#15981)
- Various possible buffer overruns have been prevented, and missed memory protection added. (PR#15990)
- `Rscript` no longer passes `--args` to R when there are no extra (“user”) arguments.
- objects like `getClass("refClass")@prototype` now `print()` and `str()` without error.
- `identical()` now also looks at the S4 bit.
- `hist(x, breaks)` is more robust in adding a small fuzz to few breaks when some are very large. (PR#15988)
- `sub()` and `gsub()` did not handle regular expressions like `"\s{2,}"` properly if the text contained NA or non-ascii elements in a UTF-8 locale. Part of this was due to a bug in the TRE library. (PR#16009)
- `RShowDoc("NEWS")` now displays the PDF version.
- Matrices and arrays with last dimension zero did not print at all or incompletely. (PR#16012)
- `plot.histogram()` and hence `hist()` now respect the `xaxs`, `yaxs` and `lab` graphics parameters. (PR#16021)
- `bw.SJ(x)` and other `bw.*()` no longer segfault when `x` contains non-finite values. (PR#16024)
- R CMD `Rd2pdf` unintentionally ignored its ‘`--os`’ option.
- The internal method of `download.file()` was not reporting file sizes and progress correctly on files larger than 2GB (inherited from `libxml2`). This is corrected for 64-bit builds (32-bit platforms may not support such files, but where possible will be supported in future versions of R).
- Work around a bug in OS X Yosemite where key environment variables may be duplicated causing issues in subprocesses. The duplicates are now removed on R startup (via `Rprofile`). (PR#16042)
- Adjust X11 auto-launch detection in `DISPLAY` on OS X to recognize latest XQuartz.

News from the Bioconductor Project

by the Bioconductor Team

The Bioconductor project provides tools for the analysis and comprehension of high-throughput genomic data. The 934 software packages available in Bioconductor can be viewed at <http://bioconductor.org/packages/release/>. Navigate packages using 'biocViews' terms and title search. Each package has an html page with a description, links to vignettes, reference manuals, and usage statistics. Start using Bioconductor and R version 3.1 with

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Install additional packages and dependencies, e.g., **GenomicAlignments**, with

```
source("http://bioconductor.org/biocLite.R")
biocLite("GenomicAlignments")
```

Upgrade installed packages with

```
source("http://bioconductor.org/biocLite.R")
biocLite()
```

Bioconductor 3.0 Release Highlights

Bioconductor 3.0 was released on 14 October 2014. It is compatible with R 3.1 and consists of 934 software packages, 219 experiment data packages, and more than 870 current annotation packages. In total, the release includes 114 new software packages and many updates and improvements to existing packages. The [release announcement](#) includes descriptions of new packages and updated NEWS files provided by package maintainers.

The variety of research areas represented by Bioconductor packages are organized (and searched) via the [biocViews](#) interface. Here we highlight a few topics covered by the new packages. Methods for differential expression analyses are offered in [ballgown](#) (assembled transcriptomes), [derfinder](#) (RNA-seq data at base-pair resolution) and [csaw](#) (differentially bound regions in ChIP-seq data). Quantitative trait loci (QTL) analysis for 1H NMR data are provided in [mQTL.NMR](#); [DOQTL](#) analyzes QTLs in multi-parent outbred populations. Copy number analysis in tumoral phenotypes and genomic focal aberrations are available in [facopy](#) and [focalCall](#), respectively. Additions to the flow cytometry family include [flowcatchR](#) with tools for analyzing in vivo microscopy imaging data of flowing blood cells, [flowCHIC](#) for analyzing flow data of microbial communities based on histogram images and [flowDensity](#) which provides tools for automated sequential gating (analogous to manual gating) based on data density. Several new packages take the pipeline approach and facilitate steps from raw data to final analysis: [groHMM](#) (GRO-seq data), [FourCSeq](#) (multiplexed 4C sequencing data), and [systemPipeR](#) (NGS applications such as RNA-Seq, ChIP-Seq, VAR-Seq).

Bioconductor is built on the mature and flexible 'Ranges' infrastructure defined in packages such as [IRanges](#), [GenomicRanges](#), [GenomicAlignments](#), and [GenomicFeatures](#). Many packages rely on the Ranges framework for interoperable, re-usable analysis; [Lawrence et al. \(2013\)](#) provide an introduction and [Lawrence and Morgan \(2014\)](#) review strategies for processing, summarizing and visualizing large genomic data.

Our collection of microarray, transcriptome and organism-specific *annotation packages* use the 'select' interface (keys, columns, keytypes) which enable programmatic access to the databases they contain. The [AnnotationHub](#) complements our traditional offerings with diverse whole genome annotations from Ensembl, ENCODE, dbSNP, UCSC, and elsewhere.

Other activities

Bioconductor offers an [Amazon Machine Image](#) optimized for running Bioconductor in the Amazon Elastic Compute Cloud (EC2). The AMI comes pre-loaded with the latest release version of R, and a subset of Bioconductor packages. The AMI can be customized by installing R packages with 'biocLite()' or system-level packages with the Ubuntu package manager 'apt-get'. Files can be transferred to the EC2 instance via scp or the Rstudio interface.

A recent addition is our collection of Bioconductor [Docker Images](#). These self-contained environments run on Linux, Windows and Mac as well as virtual machines. The containers provide convenient access to a 'fresh R session' or specific version of Bioconductor without the overhead of installing packages and dependencies. Analysis-specific images come pre-loaded with packages of a common topic such as flow, proteomics, microarray and sequencing.

New Bioconductor package contributors are encouraged to consult the [package guidelines](#) and [Package Submission](#) sections of the Bioconductor web site, and use the new [BiocCheck](#) package, in addition to R CMD check, for guidance on conforming to Bioconductor package standards.

The Bioconductor web site advertises [training and community events](#); [mailing lists](#) connect users with each other, to domain experts, and to maintainers eager to ensure that their packages satisfy the needs of leading edge approaches. Keep abreast of packages added to the 'devel' branch and other activities by following @Bioconductor on Twitter.

Bibliography

- M. Lawrence and M. Morgan. Scalable genomics with R and Bioconductor. *Statistical Science*, 29:214–226, 2014. doi: 10.1214/14-STS476. URL <http://arxiv.org/abs/1409.2864>. [p227]
- M. Lawrence, W. Huber, H. Pagès, P. Aboyoun, M. Carlson, R. Gentleman, M. T. Morgan, and V. J. Carey. Software for computing and annotating genomic ranges. *PLoS Comput Biol*, 9(8):e1003118, 08 2013. doi: 10.1371/journal.pcbi.1003118. URL <http://dx.doi.org/10.1371/journal.pcbi.1003118>. [p227]

The Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center