The R Journal

Volume 4/2, December 2012

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

Contents

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

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

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

Editorial

by Martyn Plummer

Welcome to volume 4, issue 2 of The R Journal.

Changes to the journal

Thomson Reuters has informed us that The R Journal has been accepted for listing in the Science Citation Index-Expanded (SCIE), including the Web of Science, and the ISI Alerting Service, starting with volume 1, issue 1 (May 2009). This complements the current listings by EBSCO and the Directory of Open Access Journals (DOAJ), and completes a process started by Peter Dalgaard in 2010.

Since The R Journal publishes two issues per year, the delay between acceptance and publication can sometimes be long. In July, we started putting accepted articles online, so that they are immediately accessible. If you want to see a preview of some of the articles in the next issue, go to the "Accepted Articles" page on the R Journal Web site.

A minor formatting change in this issue is the inclusion of back-links in the bibliography. Citations in The R Journal are hyper-links that will take you to the corresponding entry in the bibliography. The back-links now enable you to go back to the referring page.

In this issue

The Contributed Research Articles section of this issue opens with a trio of papers by Paul Murrell, explaining advanced graphics features of R. We then have two papers on multilevel models: Il Do Ha, Maengseok Noh, and Youngjo Lee present the **frailtyHL** package for fitting survival models, and Rense Nieuwenhuis, Manfred te Grotenhuis, and Ben Pelzer present the **influence.ME** package for diagnostics in multilevel models. We also have two papers on flexible and robust regression: Zhenghua Nie and Jeffrey Racine discuss nonparametric regression splines with the **crs** package, while John Kloke and Joseph McKean discuss rank-based regression for linear models with **Rfit**. Finally, Kayvan Sadeghi and Giovanni Marchetti show how the **ggm** package can be used to examine the statistical properties of mixed graphical models.

Changes to the editorial board

The end of the year also brings changes to the editorial board. Heather Turner is leaving the board after four years. Heather has been on the board since the first issue of R News and, in addition to being an indefatigable editor, is responsible for much of the computational infrastructure of the journal. Another departure is Bill Venables, who has been editor of Programmer's Niche since the very first issue of R News – the predecessor of The R Journal – in 2001. The last paper handled by Bill is a survey of naming conventions in R by Rasmus Bååth. We welcome Bettina Grün, who will join the editorial board in 2013. I shall be stepping down as Editor-in-Chief and will be leaving this task in the capable hands of Hadley Wickham.

What's in a Name?

The Importance of Naming grid Grobs When Drawing Plots in R

by Paul Murrell

Abstract Any shape that is drawn using the **grid** graphics package can have a name associated with it. If a name is provided, it is possible to access, query, and modify the shape after it has been drawn. These facilities allow for very detailed customisations of plots and also for very general transformations of plots that are drawn by packages based on **grid**.

Introduction

When a scene is drawn using the **grid** graphics package in R, a record is kept of each shape that was used to draw the scene. This record is called a *display list* and it consists of a list of R objects, one for each shape in the scene. For example, the following code draws several simple shapes: some text, a circle, and a rectangle (see Figure 1).

```
> library(grid)

> grid.text(c("text", "circle", "rect"),
+           x=1:3/4, gp=gpar(cex=c(3, 1, 1)))
> grid.circle(r=.25)
> grid.rect(x=3/4, width=.2, height=.5)
```

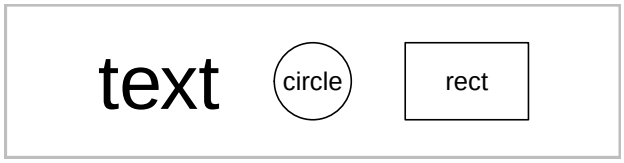


Figure 1: Some simple shapes drawn with **grid**.

The following code uses the `grid.ls()` function to show the contents of the display list for this scene. There is an object, called a *grob* (short for "graphical object"), for each shape that we drew. The output below shows what sort of shape each grob represents and it shows a name for each grob (within square brackets). In the example above, we did not specify any names, so **grid** made some up.

```
> grid.ls(fullNames=TRUE)

text[GRID.text.5]
circle[GRID.circle.6]
rect[GRID.rect.7]
```

It is also possible to explicitly name each shape that we draw. The following code does this by specifying the `name` argument in each function call (the resulting scene is the same as in Figure 1) and call `grid.ls()` again to show that the grobs on the display list now have the names that we specified.

```
> grid.text(c("text", "circle", "rect"),
+           x=1:3/4, gp=gpar(cex=c(3, 1, 1)),
+           name="leftText")
> grid.circle(r=.25, name="middleCircle")
> grid.rect(x=3/4, width=.2, height=.5,
+           name="rightRect")

> grid.ls(fullNames=TRUE)

text[leftText]
circle[middleCircle]
rect[rightRect]
```

The **grid** package also provides functions that allow us to access and modify the grobs on the display list. For example, the following code modifies the circle in the middle of Figure 1 so that its background becomes grey (see Figure 2). We select the grob to modify by specifying its name as the first argument. The second argument describes a new value for the `gp` component of the circle (in this case we are modifying the `fill` graphical parameter).

```
> grid.edit("middleCircle", gp=gpar(fill="grey"))
```

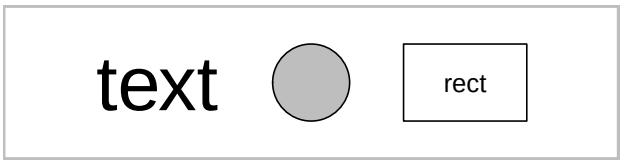


Figure 2: The simple shapes from Figure 1 with the middle circle modified so that its background is grey.

The purpose of this article is to discuss why it is useful to provide explicit names for the grobs on the **grid** display list. We will see that several positive consequences arise from being able to identify and modify the grobs on the display list.

Too many arguments

This section discusses how naming the individual shapes within a plot can help to avoid the problem of having a huge number of arguments or parameters in a high-level plotting function.

The plot in Figure 3 shows a *forest plot*, a type of plot that is commonly used to display the results of a meta-analysis. This plot was produced using the `forest()` function from the **metafor** package (Viechtbauer, 2010).

This sort of plot provides a good example of how statistical plots can be composed of a very large number of simple shapes. The plot in Figure 3 consists of

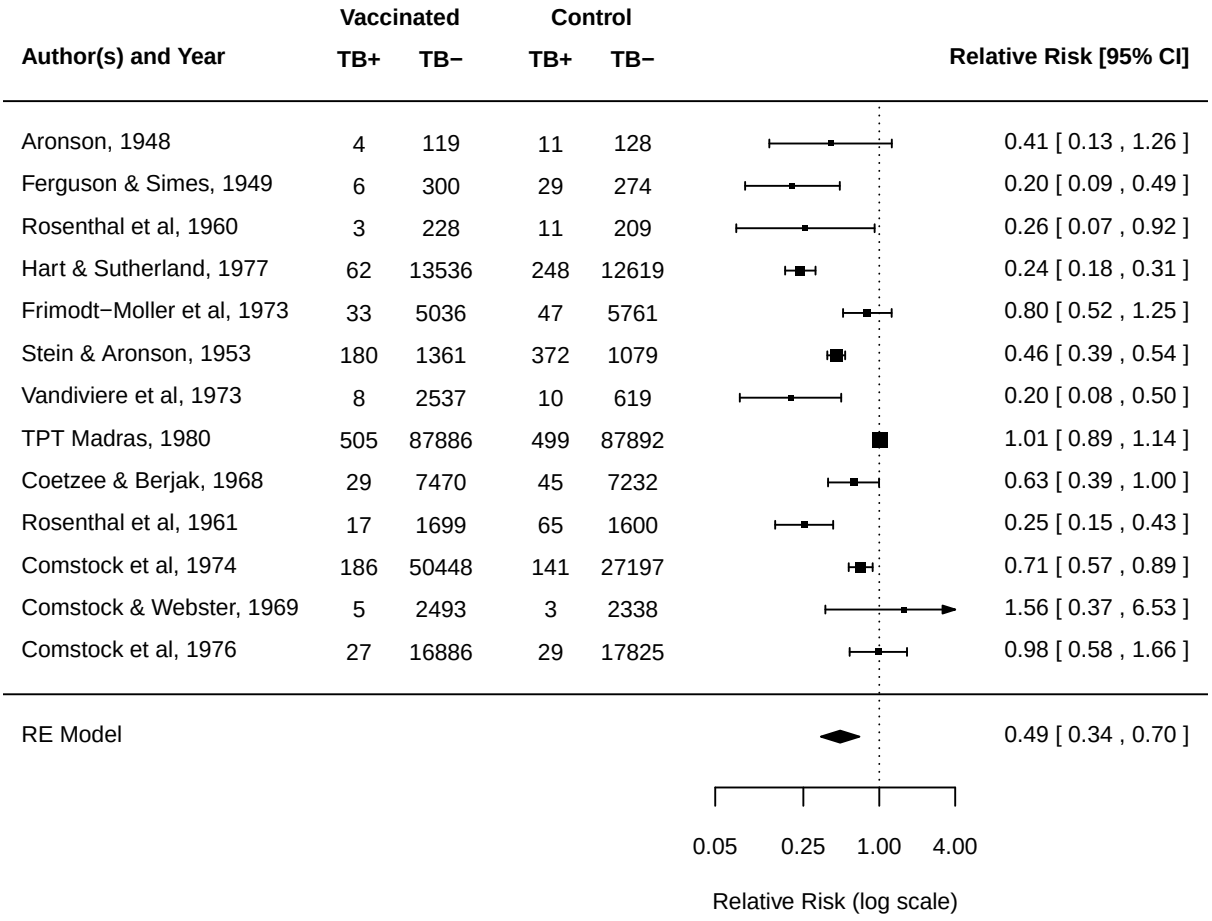


Figure 3: A forest plot produced by the `forest()` function from the **metafor** package.

many different pieces of text, rectangles, lines, and polygons.

High-level functions like `forest()` are extremely useful because, from a single function call, we can produce many individual shapes and arrange them in a meaningful fashion to produce an overall plot. However, a problem often arises when we want to *customise* individual shapes within the plot.

For example, a post to the R-help mailing list in August 2011 asked for a way to change the colour of the squares in a forest plot because none of the (thirty-three) existing arguments to `forest()` allowed this sort of control. The reply from Wolfgang Viechtbauer (author of **metafor**) states the problem succinctly:

> "The thing is, there are so many different elements to a forest plot (squares, lines, polygons, text, axes, axis labels, etc.), if I would add arguments to set the color of each element, things would really get out of hand ...
>
> ... what if somebody wants to have a different color for *one* of the squares and a different color for the other squares?"

The reality is that it is impossible to provide enough arguments in a high-level plotting function to allow for all possible modifications to the low-level shapes that make up the plot. Fortunately, an alternative is possible through the simple mechanism of providing names for all of the low-level shapes.

In order to demonstrate this idea, consider the **lattice** plot (Sarkar, 2008) that is produced by the following code and shown in Figure 4.

```
> library(lattice)

> xyplot(mpg ~ disp, mtcars)
```

This plot is simpler than the forest plot in Figure 3, but it still contains numerous individual shapes. Anyone familiar with the **lattice** package will also know that it can produce plots of much greater complexity; in general, the **lattice** package faces a very difficult problem if it wants to provide an argument in its high-level functions to control every single shape within any of its plots.

However, the **lattice** package also provides names for everything that it draws. The following code shows the contents of the **grid** display list after drawing the plot in Figure 4.

```
> grid.ls(fullNames=TRUE)

rect[plot_01.background]
text[plot_01.xlab]
text[plot_01.ylab]
segments[plot_01.ticks.top.panel.1.1]
segments[plot_01.ticks.left.panel.1.1]
text[plot_01.ticklabels.left.panel.1.1]
segments[plot_01.ticks.bottom.panel.1.1]
text[plot_01.ticklabels.bottom.panel.1.1]
segments[plot_01.ticks.right.panel.1.1]
points[plot_01.xyplot.points.panel.1.1]
rect[plot_01.border.panel.1.1]
```

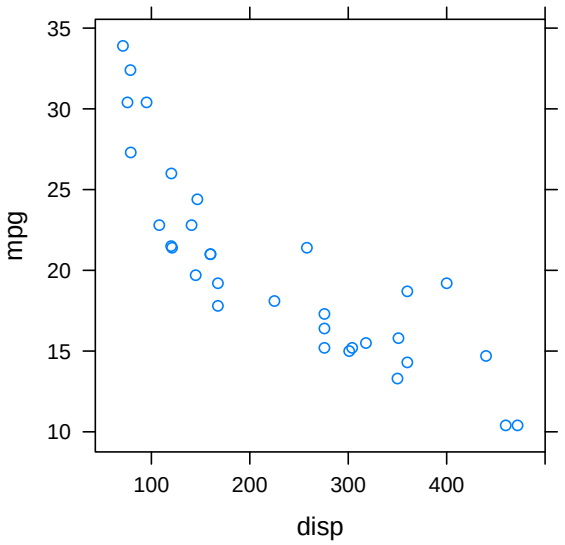


Figure 4: A simple **lattice** scatterplot.

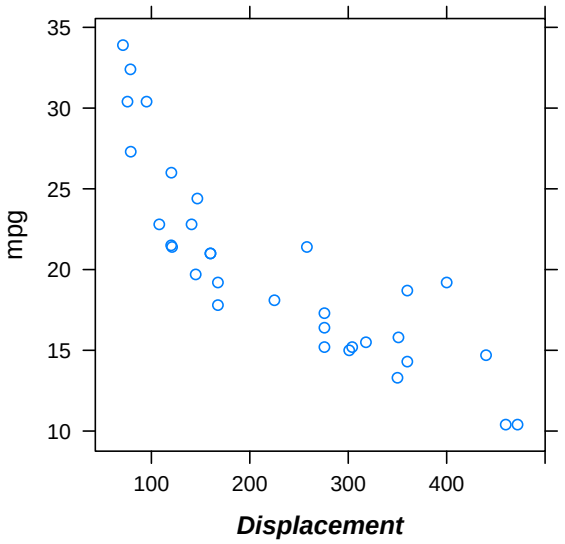


Figure 5: The **lattice** plot from Figure 4 with the x-axis modified using low-level **grid** functions.

Because everything is named, it is possible to access any component of the plot using the low-level **grid** functions. For example, the following code modifies the x-axis label of the plot (see Figure 5). We specify the component of the scene that we want to modify by giving its name as the first argument to `grid.edit()`. The other arguments describe the

changes that we want to make (a new `label` and a new `gp` setting to change the `fontface`).

```
> grid.edit("plot_01.xlab",
+           label="Displacement",
+           gp=gpar(fontface="bold.italic"))
```

That particular modification of a **lattice** plot could easily be achieved using arguments to the high-level `xyplot()` function, but the direct access to low-level shapes allows for a much wider range of modifications. For example, figure 6 shows a more complex multipanel **lattice** barchart.

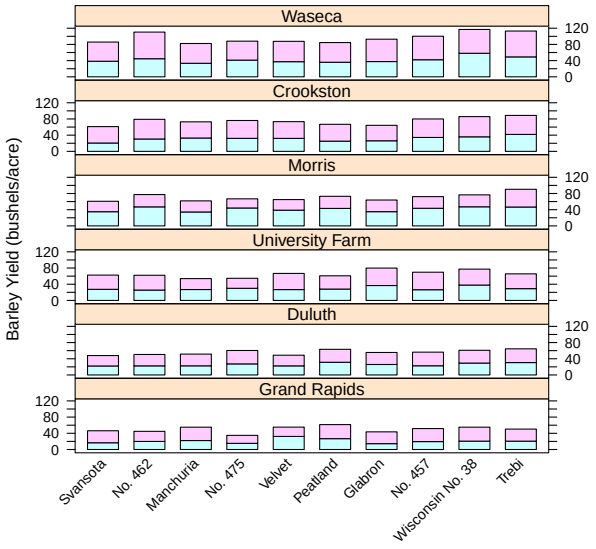


Figure 6: A complex multipanel **lattice** barchart.

This is generated by the following code

```
> barchart(yield ~ variety | site, data = barley,
+          groups = year, layout = c(1,6),
+          stack = TRUE,
+          ylab = "Barley Yield (bushels/acre)",
+          scales = list(x = list(rot = 45)))
```

There are too many individual shapes in this plot to show the full display list here, but all of the shapes have names and the following code makes use of those names to perform a more sophisticated plot modification: highlighting the sixth set of bars in each panel of the barchart (see Figure 7).

```
> grid.edit("barchart.pos.6.rect",
+           grep=TRUE, global=TRUE,
+           gp=gpar(lwd=3))
```

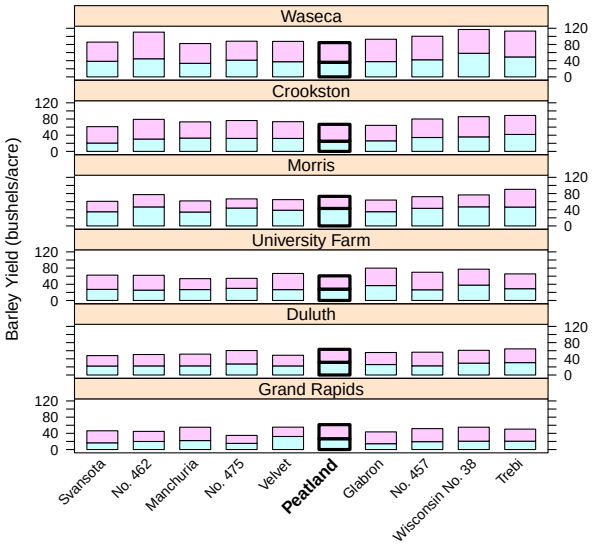


Figure 7: The barchart from Figure 6 with the sixth set of bars in each panel highlighted.

The first argument to `grid.edit()` this time is *not* the name of a specific grob. This time we have given a name *pattern*. This is indicated by the use of the `grep` argument; `grep=TRUE` means that the change will be made to a component that matches the name pattern (that was given as the first argument). The `global` argument is also set to `TRUE`, which means that this change will be made to not just the first component that matches the name pattern, but to all components that match. The `gp` argument specifies the change that we want to make (make the lines nice and thick).

It would not be reasonable to expect the high-level `barchart()` function to provide an argument that allows for this sort of customisation, but, because **lattice** has named everything that it draws, `barchart()` does not need to cater for every possible customisation. Low-level access to individual shapes can be used instead bceause individual shapes can be identified by name.

Post-processing graphics

This section discusses how naming the individual shapes within a plot allows not just minor customisations, but general transformations to be applied to a plot.

The R graphics system has always encouraged the philosophy that a high-level plotting function is only a starting point. Low-level functions have always been provided so that a plot can be customised by *adding* some *new drawing* to the plot.

The previous section demonstrated that, if every shape within a plot has a label, it is also possible to customise a plot by *modifying* the *existing shapes*

within a plot.

However, we can go even further than just modifying the existing parameters of a shape. In theory, we can think of the existing shapes within a picture as a basis for more general post-processing of the image.

As an example, one thing that we can do is to query the existing components of a plot to determine the position or size of an existing component. This means that we can position or size new drawing in relation to the existing plot. The following code uses this idea to add a rectangle around the x-axis label of the plot in Figure 4 (see Figure 8). The `grobWidth()` function is used to calculate the width of the rectangle from the width of the x-axis label. The first argument to `grobWidth()` is the name of the x-axis label grob. The `downViewport()` function is used to make sure that we draw the rectangle in the right area on the page.[1]

```
> xyplot(mpg ~ disp, mtcars)
```

```
> rectWidth <- grobWidth("plot_01.xlab")
```

```
> downViewport("plot_01.xlab.vp")
> grid.rect(width=rectWidth + unit(2, "mm"),
+           height=unit(1, "lines"),
+           gp=gpar(lwd=2),
+           name="xlabRect")
```

The display list now contains an new rectangle grob, as shown below.

```
> grid.ls(fullNames=TRUE)
```

```
rect[plot_01.background]
text[plot_01.xlab]
text[plot_01.ylab]
segments[plot_01.ticks.top.panel.1.1]
segments[plot_01.ticks.left.panel.1.1]
text[plot_01.ticklabels.left.panel.1.1]
segments[plot_01.ticks.bottom.panel.1.1]
text[plot_01.ticklabels.bottom.panel.1.1]
segments[plot_01.ticks.right.panel.1.1]
points[plot_01.xyplot.points.panel.1.1]
rect[plot_01.border.panel.1.1]
rect[xlabRect]
```

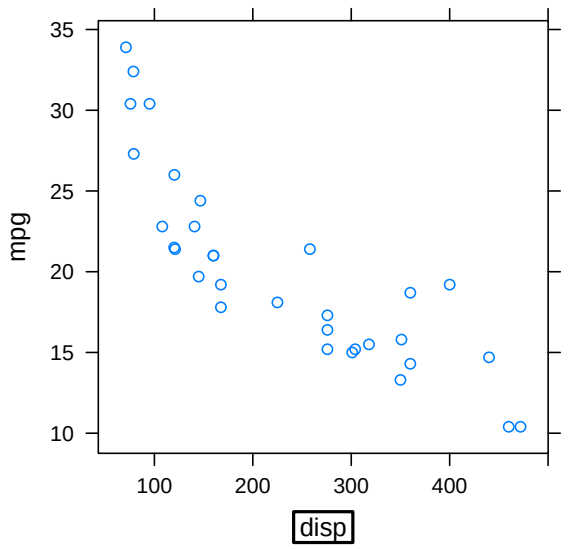


Figure 8: The **lattice** plot from Figure 4 with a rectangle added around the x-axis label.

Importantly, the new grob depends on the size of the existing x-axis label grob within the scene. For example, if we edit the x-axis label again, as below, the rectangle will grow to accommodate the new label (see Figure 9).

```
> grid.edit("plot_01.xlab",
+           label="Displacement",
+           gp=gpar(fontface="bold.italic"))
```

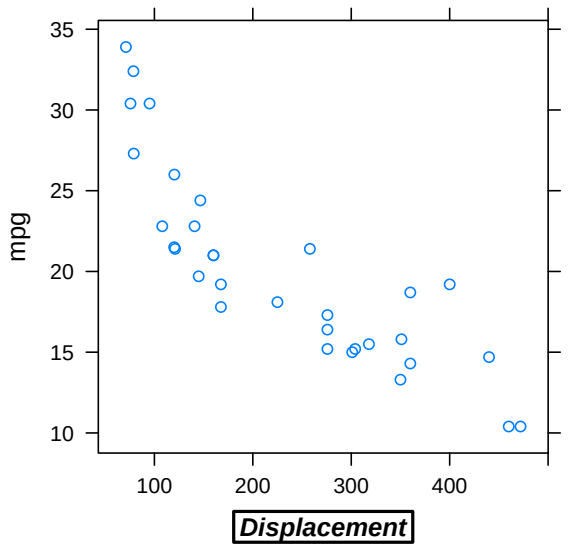


Figure 9: The **lattice** plot from Figure 4 with a rectangle added around the *modified* x-axis label.

A more extreme example of post-processing is demonstrated in the code below. In this case, we again query the existing x-axis label to determine its

[1]This `downViewport()` works because the **grid** viewports that **lattice** creates to draw its plots all have names too!

width, but this time, rather than adding a rectangle, we *replace* the label with a rectangle (in effect, we "redact" the x-axis label; see Figure 10).

```
> xyplot(mpg ~ disp, mtcars)

> xaxisLabel <- grid.get("plot_01.xlab")
> grid.set("plot_01.xlab",
+          rectGrob(width=grobWidth(xaxisLabel) +
+                           unit(2, "mm"),
+                   height=unit(1, "lines"),
+                   gp=gpar(fill="black"),
+                   name="plot_01.xlab"))
```

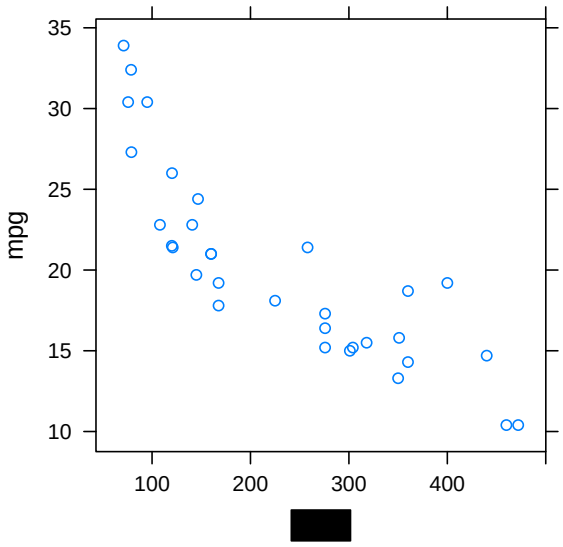


Figure 10: The **lattice** plot from Figure 4 with the x-axis label redacted (replaced with a black rectangle).

The display list now consists of the same number of grobs as in the original plot, but now the grob named `"plot_01.xlab"` is a *rectangle* instead of text (see the second line of the output below).

```
> grid.ls(fullNames=TRUE)

rect[plot_01.background]
rect[plot_01.xlab]
text[plot_01.ylab]
segments[plot_01.ticks.top.panel.1.1]
segments[plot_01.ticks.left.panel.1.1]
text[plot_01.ticklabels.left.panel.1.1]
segments[plot_01.ticks.bottom.panel.1.1]
text[plot_01.ticklabels.bottom.panel.1.1]
segments[plot_01.ticks.right.panel.1.1]
points[plot_01.xyplot.points.panel.1.1]
rect[plot_01.border.panel.1.1]
```

The artificial examples shown in this section so far have been deliberately simple in an attempt to make the basic concepts clear, but the ideas can be applied on a much larger scale and to greater effect. For example, the **gridSVG** package (Murrell, 2011) uses these techniques to transform static R plots into dynamic and interactive plots for use in web pages. It has functions that modify existing grobs on the **grid** display list to add extra information, like hyperlinks and animation, and it has functions that transform each grob on the **grid** display list to SVG code. The following code shows a simple demonstration where the original **lattice** plot is converted to an SVG document with a hyperlink on the x-axis label. Figure 11 shows the SVG document in a web browser.

```
> xyplot(mpg ~ disp, mtcars)

> library(gridSVG)

> url <-
+   "http://www.mortality.org/INdb/2008/02/12/8/document.pdf"

> grid.hyperlink("plot_01.xlab", href=url)
> gridToSVG("xyplot.svg")
```

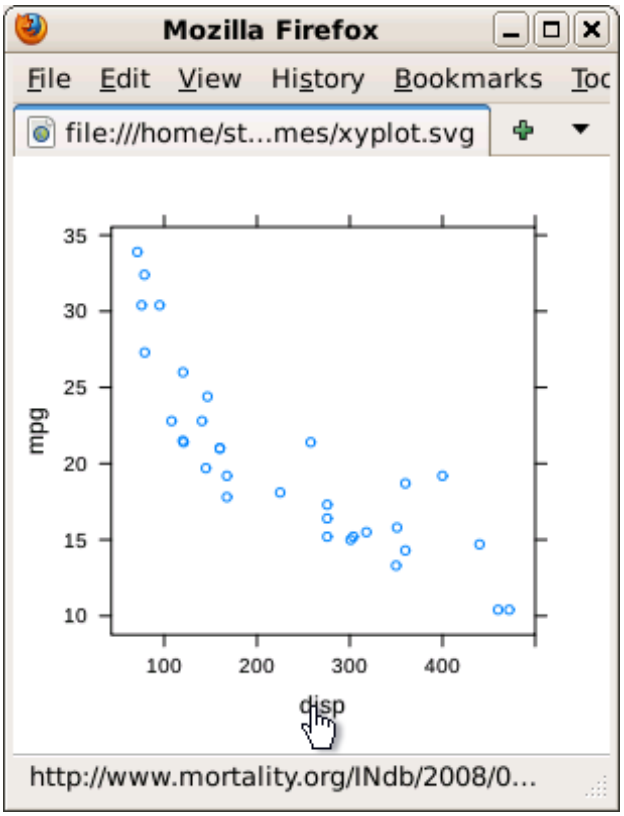


Figure 11: The **lattice** plot from Figure 4 transformed into an SVG document with a hyperlink on the x-axis label.

The significant part of that code is the first argument in the call to the `grid.hyperlink()` function, which demonstrates the ability to specify a plot component by name.

More sophisticated embellishments are also possible with **gridSVG** because the names of plot components are exported to SVG code as `id` attributes of the corresponding SVG elements. This facilitates the development of javascript code to allow user interaction with the SVG plot and allows for the possibility of CSS styling of the SVG plot.

Naming schemes

The basic message of this article is straightforward: name everything that you draw with **grid**. However, deciding what names to use—deciding on a *naming scheme*—is not necessarily so easy.

The approach taken in the **lattice** package is to attempt to reflect the structure of the plot in the naming scheme. For example, everything that is drawn within a panel region has the word `"panel"` in its name, along with a suffix of the form *i.j* to identify the panel row and column.

The decision may be made a lot easier if a plot is drawn from gTrees rather than simple grobs, because the gTrees reflect the plot structure already and names for individual components can be chosen to reflect just the "local" role of each plot component. The naming scheme in the **ggplot2** package (Wickham, 2009) is an example of this approach.

In addition to the code developer deciding on a naming scheme, the code user also faces the problem of how to "discover" the names of the components of a plot.

From the developer side, there is a responsibility to document the naming scheme (for example, the **lattice** naming scheme is described on the packages's R-Forge web site[2]). It may also be possible to provide a function interface to assist in constructing the names of grobs (for example, the `trellis.grobname()` function in **lattice**).

From the user side, there are tools that help to display the names of grobs in the current scene. This article has demonstrated the `grid.ls()` function, but there is also a `showGrob()` function, and the **gridDebug** package (Murrell and Ly., 2011) provides some more tools.

Caveats

The examples used for demonstrations in this article are deliberately simplified to make explanations clearer. This section addresses two complications that have not been raised previously.

One issue is that, while each call to a **grid** drawing function produces exactly one grob, a single call to a drawing function may produce *more than one* shape in the scene. In the very first example in this article (Figure 1), the call to `grid.circle()` creates one circle grob and draws one circle.

```
> grid.circle(r=.25, name="middleCircle")
```

The call to `grid.text()` also creates only one text grob, but it draws *three* pieces of text.

```
> grid.text(c("text", "circle", "rect"),
+           x=1:3/4, gp=gpar(cex=c(3, 1, 1)),
+           name="leftText")
```

[2] http://lattice.r-forge.r-project.org/documentation.php

Modifying this text grob is slightly more complex because there are three locations and three sets of graphical parameter settings for this single grob. For example, if we modify the text grob and supply a single `cex` setting, that is applied to all pieces of text (see Figure 12).

```
> grid.edit("leftText", gp=gpar(cex=2))
```

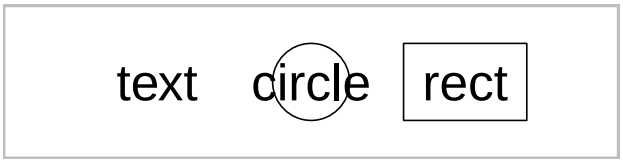


Figure 12: The simple shapes from Figure 1 with the text grob modified using a single `cex` value.

If we want to control the `cex` for each piece of text separately, we must provide three new settings (see Figure 13).

```
> grid.edit("leftText", gp=gpar(cex=c(1, 2, 3)))
```

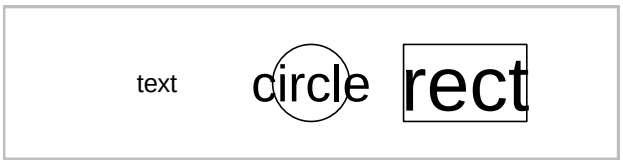


Figure 13: The simple shapes from Figure 1 with the text grob modified using three distinct `cex` values.

Another topic that has not been mentioned is **grid** viewports. This is because, although **grid** viewports can also be named, they cannot be edited in the same way as grobs (the names are only used for navigation between viewports). Furthermore, **grid** does not allow the `vp` slot on a grob to be modified and the `name` slot on grobs is also out of bounds. These limitations are imposed because the consequences of allowing modifications are either nonsensical or too complex to currently be handled by **grid**.

Discussion

In summary, if we specify an explicit name for every shape that we draw using **grid**, we allow low-level access to every grob within a scene. This allows us to make very detailed customisations to the scene, without the need for long lists of arguments in high-level plotting functions, and it allows us to query and transform the scene in a wide variety of ways.

An alternative way to provide access to individual shapes within a plot is to allow the user to simply select shapes on screen via a mouse. How does this compare to a naming scheme?

Selection using a mouse works well for some sorts of modifications (see, for example, the **playwith** package; Andrews, 2010), but providing access to individual shapes by name is more efficient, more general, and more powerful. For example, if we write code to make modifications, referencing grobs by name, we have a record of what we have done, we can easily automate large numbers of modifications, we can share our modification techniques, and we can express more complex modifications (like "highlight every sixth bar").

Another alternative way to provide detailed control over a scene is simply to modify the original R code that drew the scene. Why go to the bother of naming grobs when we can just modify the original R code?

If we have written the original code, then modifying the original code may be the right approach. However, if we draw a plot using someone else's code (for example, if we call a **lattice** function), we do not have easy access to the code that did the drawing. Even though it is possible to see the code that did the drawing, understanding it and then modifying it may require a considerable effort, especially when that code is of the size and complexity of the code in the **lattice** package.

A parallel may be drawn between this idea of naming every shape within a scene and the general idea of *markup*. In a sense, what we are aiming to do is to provide a useful label for each meaningful component of a scene. Given tools that can select parts of the scene based on the labels, the scene becomes a "source" that can be transformed in many different ways. When we draw a scene in this way, it is not just an end point that satisfies our own goals. It also creates a resource that others can make use of to produce new resources. When we write code to draw a scene, we are not only concerned with producing an image on screen or ink on a page; we also allow for other possible uses of the scene in ways that we may not have anticipated.

Acknowledgements

Thanks to Wolfgang Viechtbauer for useful comments on an early draft of this article and to the anonymous referees for numerous useful suggestions for improvements.

Bibliography

F. Andrews. *playwith: A GUI for interactive plots using GTK+*, 2010. URL http://CRAN.R-project.org/package=playwith. R package version 0.9-53. [p12]

P. Murrell. *gridSVG: Export grid graphics as SVG*, 2011. URL http://CRAN.R-project.org/package=gridSVG. R package version 0.7-0. [p10]

P. Murrell and V. Ly. *gridDebug: Debugging Grid Graphics*, 2011. URL http://r-forge.r-project.org/projects/griddebug/. R package version 0.2. [p11]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p7]

W. Viechtbauer. Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, 36(3):1–48, 2010. URL http://www.jstatsoft.org/v36/i03/. [p5]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p11]

Paul Murrell
Department of Statistics
The University of Auckland
New Zealand
paul@stat.auckland.ac.nz

It's Not What You Draw, It's What You Don't Draw

Drawing Paths with Holes in R Graphics

by Paul Murrell

Abstract The R graphics engine has new support for drawing complex paths via the functions `polypath()` and `grid.path()`. This article explains what is meant by a complex path and demonstrates the usefulness of complex paths in drawing non-trivial shapes, logos, customised data symbols, and maps.

One of the design goals of the R graphics system is to allow fine control over the small details of plots. One way that the R graphics system does this is by providing access to low-level generic graphics facilities, such as the ability to draw basic shapes and the ability to control apparently esoteric, but still useful, features of those shapes, such as the line end style used for drawing lines.

In R version 2.12.0, another low-level graphics facility was added to R: the ability to draw *complex paths* (not just polygons).

This article describes this new facility and presents some examples that show how complex paths might be useful.

Drawing paths with holes

The concept of a *path* is similar to the concept of a polygon: a path is defined by a series of (x,y) locations that describe the boundary of the path.

For example, the following code defines a set of (x,y) locations that describe a simple triangle.

```
> x <- c(.1, .5, .9)
> y <- c(.1, .8, .1)
```

A triangle can be drawn from these locations using either the `polypath()` function from the **graphics** package or, as shown below and in Figure 1, using the `grid.path()` function from the **grid** package.

```
> library(grid)
```

```
> grid.path(x, y, gp=gpar(fill="grey"))
```

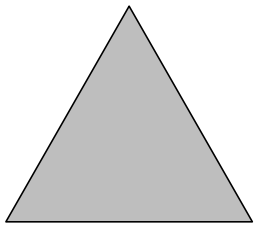


Figure 1: A triangle drawn by the `grid.path()` function from a set of three (x,y) locations.

As for any basic shape, it is possible to control the colour and thickness of the path border and the colour used to fill the interior of the path.

We can also provide more than one set of (x,y) locations when drawing a path. The following code provides an example, defining a new set of six locations along with an `id` vector that can be used to break the locations into two groups of three.

```
> x <- c(.1, .5, .9,
+        .1, .2, .3)
> y <- c(.1, .8, .1,
+        .7, .6, .7)
> id <- rep(1:2, each=3)
```

```
> cbind(x, y, id)
```

```
      x   y id
[1,] 0.1 0.1  1
[2,] 0.5 0.8  1
[3,] 0.9 0.1  1
[4,] 0.1 0.7  2
[5,] 0.2 0.6  2
[6,] 0.3 0.7  2
```

These locations can be used to describe a path that consists of two distinct triangles. The following code draws such a path using `grid.path()`. The `id` argument is used to identify distinct groups of locations when using `grid.path()`.[1] Figure 2 shows the result of drawing this path.

```
> grid.path(x, y, id=id,
+           gp=gpar(fill="grey"))
```

[1] When using `polypath()`, `NA` values must be inserted between distinct groups of (x,y) values.

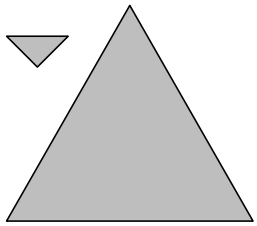


Figure 2: Two triangles drawn by the `grid.path()` function from a set of six (x,y) locations broken into two groups of three locations.

This output looks exactly the same as the output we would get from drawing the two groups of locations as polygons, using `grid.polygon()` or `polygon()`, but conceptually there is a difference because the path treats the two groups of locations as defining a single shape. We can see the difference more clearly if we move the smaller triangle so that it lies within the larger triangle (see Figure 3).

```
> x <- c(.1, .5, .9,
+        .4, .5, .6)
> y <- c(.1, .8, .1,
+        .5, .4, .5)

> grid.path(x, y, id=id,
+           gp=gpar(fill="grey"))
```

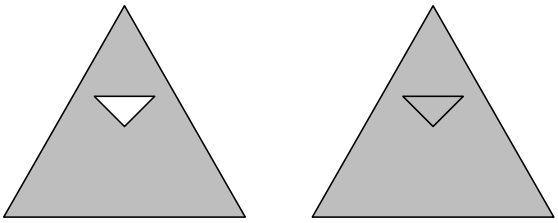


Figure 3: On the left is a path drawn by the `grid.path()` function where the boundary of the path consists of two distinct triangles (one within the other). The result is a single shape with a hole in it. On the right is the result of drawing the two boundaries with the `grid.polygon()` function, which treats the boundaries as two separate shapes. In this case, the smaller triangle is drawn (filled in) on top of the larger triangle.

This example demonstrates that the two triangles together define a single shape, which is a triangular region with a triangular hole in it. The interior of the shape—the area that is shaded—does *not* include the region within the smaller triangle.

Fill rules

There are two ways to determine the interior of a path like this. The default is called the *non-zero winding rule*. This draws an imaginary straight line and looks at where the straight line intersects the boundary of the shape. A count is made of how many times the boundary is running left-to-right at the intersection and how many times the boundary is running right-to-left; if the two counts are the same then we are outside the shape and if the counts are different, we are inside the shape.

To see this more clearly, Figure 4 uses lines with arrows to show the directions on the boundaries of the path from Figure 3.

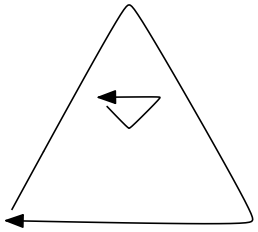


Figure 4: The direction of the boundaries for the path in Figure 3.

The outer triangle boundary is clockwise and the inner triangle boundary is anti-clockwise, so, using the non-zero winding rule, the region within the inner triangle is actually outside the path. A straight line from inside the inner triangle to outside the outer triangle intersects two boundaries, one going right-to-left and one going left-to-right.

To further demonstrate this rule, the following code defines a more complex path, this time consisting of three triangles: one large clockwise triangle, with two smaller triangles inside, one clockwise and one anti-clockwise.

```
> x <- c(.1, .5, .9,
+        .4, .5, .6,
+        .4, .6, .5)
> y <- c(.1, .8, .1,
+        .5, .4, .5,
+        .3, .3, .2)
> id <- rep(1:3, each=3)
```

Figure 5 shows a diagram of the boundary directions and the result of drawing this path. Because the second smaller triangle is clockwise, the region inside that triangle is still part of the interior of the path, according to the non-zero winding rule.

```
> grid.path(x, y, id=id,
+           gp=gpar(fill="grey"))
```

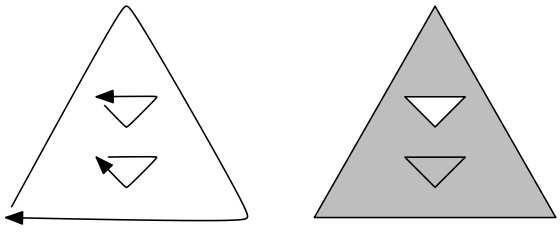


Figure 5: A path where the boundary consists of three triangles (two smaller ones within one larger one). The diagram on the left shows the direction of the boundaries for the path. On the right, the path is drawn by the `grid.path()` function, with the interior of the path determined using the non-zero winding rule.

The other rule for determining the interior of a path is called the *even-odd* rule. This just draws an imaginary straight line through the shape and counts how many times the straight line crosses the boundary of the shape. Each time a boundary is crossed, we toggle between outside and inside the shape.

The following code draws the same path as in Figure 5, but uses the even-odd rule to determine the shape's interior. This time, the result is a larger triangle with *two* smaller triangular holes punched out of it (see Figure 6).

```
> grid.path(x, y, id=id,
+           rule="evenodd",
+           gp=gpar(fill="grey"))
```

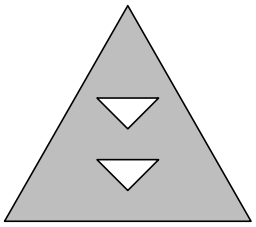


Figure 6: The path from Figure 5 drawn using the even-odd fill rule.

The SVG language specification contains a nice simple explanation and demonstration of these fill rules; see `http://www.w3.org/TR/SVG/painting.html#FillRuleProperty`.

Applications

So what can these complex paths be used for? The possibilities are endless, but this section describes a couple of concrete examples. The R code for these examples can be obtained from the online resources that accompany this article.[2]

A trivial observation is that complex paths allow us to draw complex shapes. The triangle with triangular holes from the previous section is an example of a complex shape; it is not possible to describe this shape as a simple polygon.

Another way that paths can be useful for drawing complex shapes is that they allow us to combine several simpler shapes to construct a more complex whole. Figure 7 shows an example of this, where the overall shape shape has a very complex outline, but it can be constructed as a path simply by combining circles and triangles.

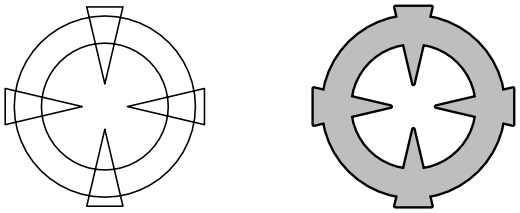


Figure 7: A complex shape constructed from simple shapes combined together to make a path.

Figure 8 shows how this shape might be used to dramatically highlight a point of interest within a graph (in this case, to bring attention to the data for the *Ferrari Dino* in the `mtcars` data set).

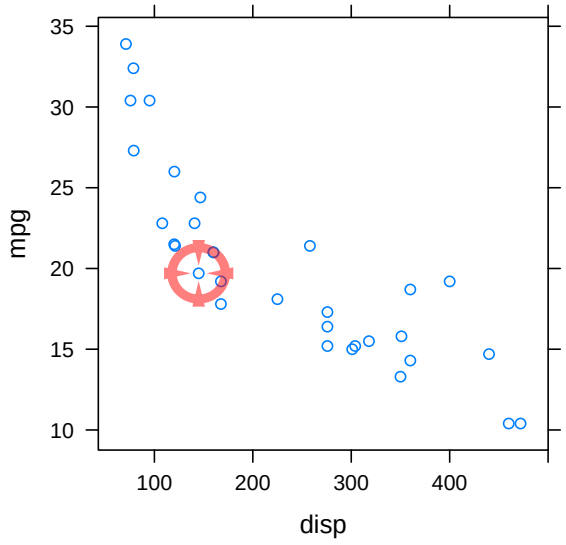


Figure 8: A plot with a complex path used to highlight a special point of interest.

Another situation where the ability to draw com-

[2] `http://www.stat.auckland.ac.nz/~paul/R/Paths/`

plex paths can be useful is if we are trying to draw a shape that someone else has created. For example, we might want to draw the logo of a company or an organisation as a label on a plot.

Figure 9 shows the GNU logo. This image consists of a single complex path, so we must be able to draw such paths in order to render it correctly.

Figure 9: A complex path that describes the GNU logo.

Figure 10 shows the GNU logo being used as a background watermark for a **lattice** barchart (Sarkar, 2008).

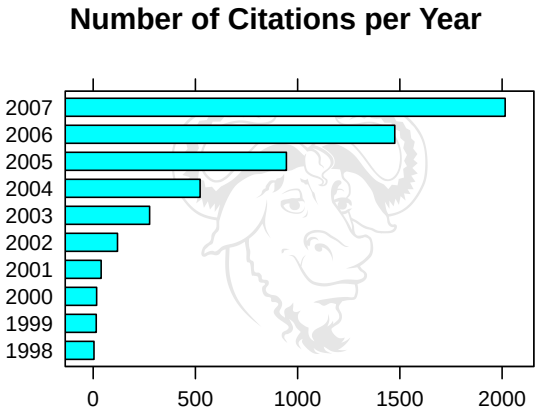


Figure 10: A plot with the GNU logo from Figure 9 as a background watermark.

Another way that we might use external complex shapes is as data symbols on a plot. Figure 11 shows a bus icon. Again, this bus icon is a single path so it must be drawn using `grid.path()` or `polypath()` in order for it to render correctly.

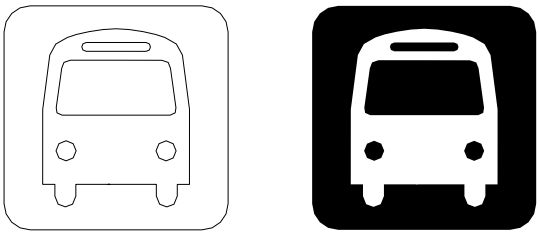


Figure 11: A path that describes a bus icon.

Figure 12 shows this bus icon being used as data symbols on a **lattice** scatterplot of daily bus ridership data.

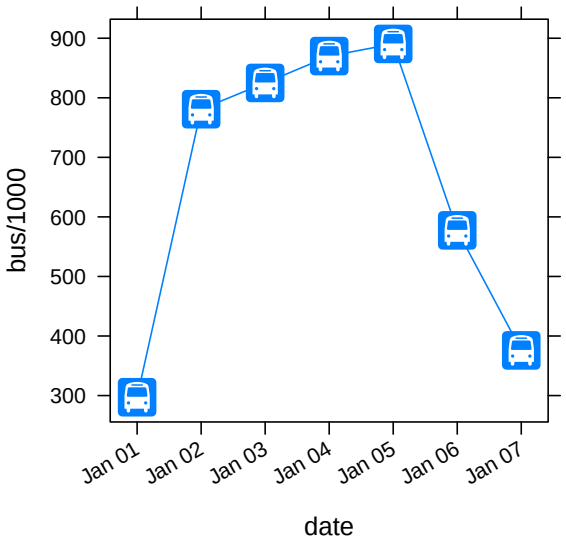


Figure 12: A plot with the bus icon from Figure 11 used as a data symbol.

Another general area where complex paths arise is the drawing of maps. The outline of a country's borders or coastline represents a set of (x,y) coordinates, often in a very complicated arrangement. One situation where it can be useful to treat the map outline as a path is the case where a country contains a lake; the lake can be thought of as a hole in the country shape. Things can get quite complicated if the lake then contains an island, and the island has a lake, and so on. If the map is treated as a path to fill then all of these cases are dealt with quite easily.

Figure 13 shows a map of part of the South Island of New Zealand. The lake in the lower right quadrant of this map is Lake Te Anau and at the base of one of the westerly spurs of this lake is an island. This map outline has been drawn as a path with a green fill colour used to indicate land area and an appropriate fill rule ensures that the lake is not filled in, but the island on the lake is.

Figure 13: A map showing an island in a lake on an island.

Although R provides many low-level graphics facilities, such as the ability to draw complex paths, there are still some basic tricks that it does not yet support. One example is the ability to clip output to an arbitrary region on the page (it is currently only possible to clip to rectangular regions with R).

Sometimes, a missing feature like this can be worked around by making inventive use of the existing facilities. Figure 14 shows an example of this, where a contour of earthquake events has been overlaid on a map of New Zealand, but the contours are only visible over land areas.

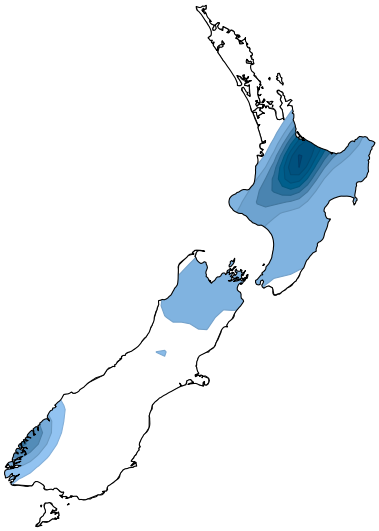


Figure 14: A map with an overlaid contour. A path has been used to obscure the contour where it does not overlap with land.

This result was achieved using complex paths (see Figure 15). The starting point is the entire contour overlaid on a New Zealand map (left). A path is constructed from the New Zealand coastline (middle), and then a bounding rectangle is added to the path (right). This combined path allows us to fill a region that is everything outside of the New Zealand coastline and that can be drawn on top of the original image to obscure those parts of the contour that are not over land.

Caveats

The `polypath()` and `grid.path()` functions are only supported on the `pdf()`, `postscript()`, `x11(type="cairo")`, `windows()`, and `quartz()` graphics devices (and associated raster formats).

These functions are not supported on `x11(type="Xlib")`, `xfig()`, or `pictex()` and support is not guaranteed on graphics devices provided by extension packages.

Summary

There are new functions, `polypath()` and `grid.path()` for drawing complex paths, including paths with holes, in R graphics output. These functions can be useful for drawing non-trivial shapes, logos, custom data symbols, and maps.

Acknowledgements

The following material and data were used in this article:

- The GNU logo was created by Aurelio A. Heckert and is available from `http://www.gnu.org/graphics/heckert_gnu.html`.
- The bus icon was created by the Geographics Unit, School of Environment, The University of Auckland.
- Both the GNU logo and the bus icon shape information were imported into R and drawn using the **grImport** package (Murrell, 2009).
- The bus data were obtained from the City of Chicago Data Portal `http://data.cityofchicago.org/Transportation/CTA-Ridership-Daily-Boarding-Totals/`.
- The detailed map of the South Island of New Zealand came from the Global Self-consistent, Hierarchical, High-resolution Shoreline Database (version 2.0; Wessel and Smith, 1996) and was loaded into R using the **maptools** package (Lewin-Koh and Bivand, 2011).
- The earthquake data came from the GeoNet Project: `http://www.geonet.org.nz/`

Figure 15: A map with a path used to obscure unwanted drawing.

- The New Zealand coastline information for Figures 14 and 15 came from the **maps** package (Brownrigg and Minka, 2011).

Many thanks also to the anonymous reviewers who suggested several useful improvements to this article.

Bibliography

R. Brownrigg and T. P. Minka. *maps: Draw Geographical Maps*, 2011. URL http://CRAN.R-project.org/package=maps. R package version 2.1-6. [p18]

N. J. Lewin-Koh and R. Bivand. *maptools: Tools for reading and handling spatial objects*, 2011. URL http://CRAN.R-project.org/package=maptools. R package version 0.8-6. [p17]

P. Murrell. Importing vector graphics: The grImport package for R. *Journal of Statistical Software*, 30(4): 1–37, 2009. URL http://www.jstatsoft.org/v30/i04/. [p17]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p16]

P. Wessel and W. H. F. Smith. A global self-consistent, hierarchical, high-resolution shoreline database. *Journal of Geophysical Research*, pages 8741–8743, 1996. [p17]

Paul Murrell
Department of Statistics
The University of Auckland
New Zealand
paul@stat.auckland.ac.nz

Debugging grid Graphics

by Paul Murrell and Velvet Ly

Abstract A graphical scene that has been produced using the **grid** graphics package consists of grobs (graphical objects) and viewports. This article describes functions that allow the exploration and inspection of the grobs and viewports in a **grid** scene, including several functions that are available in a new package called **gridDebug**. The ability to explore the grobs and viewports in a **grid** scene is useful for adding more drawing to a scene that was produced using **grid** and for understanding and debugging the **grid** code that produced a scene.

Introduction

The **grid** graphics package for R contains features that are intended to assist in the creation of flexible, complex graphical scenes, such as the plots that are produced by **lattice** (Sarkar, 2008) and **ggplot2** (Wickham, 2009).

Two particularly important features are *viewports*, which represent rectangular regions on the page for drawing into, and *grobs*, which represent shapes that have been drawn onto the page.

To illustrate these **grid** concepts, the following code draws a simple scene consisting of a narrow "strip" region atop a larger "panel" region, with a rectangle boundary drawn for each region and the top region shaded grey (see Figure 1).

```
> library(grid)
```

```
> stripVP <- viewport(y=1,
+                     height=unit(1, "lines"),
+                     just="top",
+                     name="stripvp")
> panelVP <- viewport(y=0,
+                     height=unit(1, "npc") -
+                            unit(1, "lines"),
+                     just="bottom",
+                     name="panelvp")
```

```
> pushViewport(stripVP)
> grid.rect(gp=gpar(fill="grey80"),
+           name="striprect")
> upViewport()
> pushViewport(panelVP)
> grid.rect(name="panelrect")
> upViewport()
```

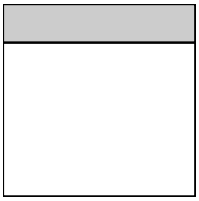


Figure 1: A scene consisting of two viewports, with a rectangle drawn in each.

One benefit that accrues from using viewports to draw the scene in Figure 1 is that, once the scene has been drawn, the viewports can be revisited to add further drawing to the scene. For example, the following code revisits the "strip" region and adds a text label (see Figure 2).

```
> downViewport("stripvp")
> grid.text("strip text", name="striptext")
> upViewport()
```

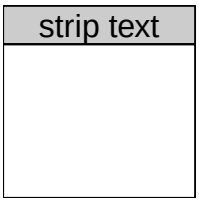


Figure 2: The scene from Figure 1 with text added to the top viewport.

One benefit that accrues from the fact that **grid** creates grobs representing the shapes in a scene is that, after the scene has been drawn, it is possible to modify elements of the scene. For example, the following code modifies the text that was just drawn in the strip region so that it is dark green, italic, and in a serif font (see Figure 3).

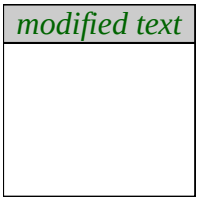


Figure 3: The scene from Figure 2 with the text modified to be dark green, italic, and serif.

```
> grid.edit("striptext",
+           label="modified text",
+           gp=gpar(col="darkgreen",
+             fontface="italic",
+             fontfamily="serif"))
```

The following code shows that it is also possible to remove objects from a scene — this returns the scene to its original state (Figure 1) by removing the text that we had added above.

```
> grid.remove("striptext")
```

The importance of names

The ability to navigate within viewports in a scene and the ability to modify grobs within a scene both depend upon being able to unambiguously specify a particular viewport or grob.

All viewports and grobs have a *name*, so specifying a particular viewport or grob is simply a matter of specifying the relevant viewport name or grob name.

In the simple example above, this is not a difficult task because we have the code that created the scene so we can see the names that were used. However, when a scene has been generated by someone else's code, for example, a call to a **lattice** plotting function, it may not be very easy to determine the name of a viewport or grob.[1]

Pity the developer

Problems can also arise when we want to develop new functions that draw scenes using **grid**. In this case, knowing the names of viewports and grobs is not the problem because we have created the names. Instead, the problem is knowing where on the page the viewports and grobs have ended up. The result of running error-ridden **grid** code can be a confusing jumble of drawing output. In this case, it is useful to be able to identify where on the page a particular viewport or grob has been drawn.

Pity the student

Even when the author of a piece of **grid** code knows exactly what the code is doing, and the code is behaving correctly, it can be difficult to convey to other people the relationship between the **grid** code and the output that it produces on a page. This is another situation where it can be useful to provide a visual cue about the location on the page of abstract concepts such as viewports and grobs and the relationships between them.

This article describes a number of functions that are provided by the **grid** package and the **gridDebug** package (Murrell and Ly, 2011) to help identify what viewports and grobs have been used to create a scene and track exactly where each viewport and grob has been drawn on the page. These functions will be introduced in the following sections using the very simple **grid** scene that was described above. These introductions are then followed by a section that looks at some more complex **grid** scenes in order to demonstrate more sophisticated uses of the functions, plus some alternative tools.

The `grid.ls()` function

A simple listing of the names of all grobs in a scene can be produced using the `grid.ls()` function. For example, the following code lists the grobs in Figure 1, which consists of just two rectangles called `"striprect"` and `"panelrect"`

```
> grid.ls()

striprect
panelrect
```

The `grid.ls()` function can also be used to list viewports in the current scene, via the `viewports` argument and the `fullNames` argument can be specified to print further information in the listing so that it is easier to distinguish viewports from grobs. The following code produces a more complete listing of the scene from Figure 1 with both viewports and grobs listed. Notice that the names are indented to reflect the fact that some viewports are nested within others and also to reflect the fact that the grobs are drawn within different viewports.

```
> grid.ls(viewports=TRUE, fullNames=TRUE)

viewport[ROOT]
  viewport[stripvp]
    rect[striprect]
    upViewport[1]
  viewport[panelvp]
    rect[panelrect]
    upViewport[1]
```

This function is useful for at least viewing the names of all grobs and viewports in a scene and it gives some indication of the structure of the scene. Even for a complex scene, such as a **lattice** multipanel conditioning plot it is possible, if a little tedious, to identify important components of the scene.

The `showGrob()` function

The `showGrob()` function displays the names of the grobs in a scene by labelling them on the current scene. By default, a semitransparent rectangle is drawn to show the extent of each grob and the name of the grob is drawn within that rectangle. For example, the following code labels the grobs in the simple scene from Figure 1. The resulting labelled scene is shown in Figure 4 — there are two rectangles called `"striprect"` and `"panelrect"`.

```
> showGrob()
```

[1] The **lattice** package does provide some assistance in the form of the `trellis.vpname()` and `trellis.grobname()` functions.

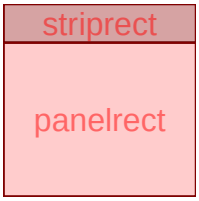


Figure 4: The scene from Figure 1 with labelling added by the `showGrob()` function to show the locations and names of the grobs used to draw the scene.

In more complex scenes, it is common for several grobs to overlap each other so that this sort of labelling becomes very messy. Later sections will demonstrate how to cope with that complexity using other functions and other arguments to the `showGrob()` function.

The `showViewport()` function

The `showViewport()` function performs a similar task to `showGrob()` except that it labels the viewports in a scene. Again, the labelling consists of a semitransparent rectangle and the name of the viewport. For example, the following code labels the viewports in the scene from Figure 1, which has a narrow viewport called `"stripvp"` on top and a larger viewport called `"panelvp"` below.

```
> showViewport()
```

Figure 5: The scene from Figure 1 with labelling added by the `showViewport()` function to show the locations and names of the viewports that were used to draw the scene.

In more complex scenes, it is common for viewports to overlap each other, so the default output from `showViewport()` is less legible. Later sections will describe solutions to this problem using further arguments to `showViewport()` as well as different debugging functions.

The gridDebug package

The **gridDebug** package provides some additional tools for debugging **grid** output. The `gridTree()` function draws a *scene graph* from a **grid** scene, using the **graph** and **Rgraphviz** packages (Gentleman et al., 2010; Gentry et al., 2010), via the **gridGraphviz** package (Murrell, 2011). This is a node-and-edge graph that contains a node for each grob and each viewport in the current **grid** scene. The graph has an edge from each child viewport to its parent viewport and an edge from each grob to the viewport within which the grob is drawn. The nodes are labelled with the name of the corresponding grobs and viewports. For example, the following code produces a scene graph for the simple scene in Figure 1. The scene graph is shown in Figure 6.

```
> library(gridDebug)

> gridTree()
```

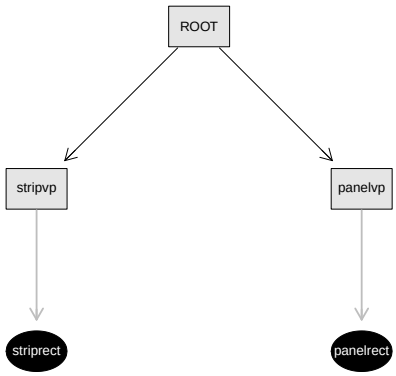


Figure 6: A node-and-edge graph of the scene from Figure 1. Both viewports are direct descendants of the `ROOT` viewport and one grob is drawn in each viewport.

This graph shows that the two viewports have both been pushed directly beneath the `ROOT` viewport (they are siblings) and that each grob has been drawn in a separate viewport.

One advantage of this function is that it is unaffected by overlapping grobs or viewports. The main downside is that node labels become very small as the scene becomes more complex.

More complex scenes

We will now consider a more complex scene and look at how the various debugging functions that have just been described can be adapted to cope with the additional complexity. As an example, we will look at a plot produced by the `histogram()` function from the **lattice** package (see Figure 7).

```
> library(lattice)

> histogram(faithful$eruptions)
```

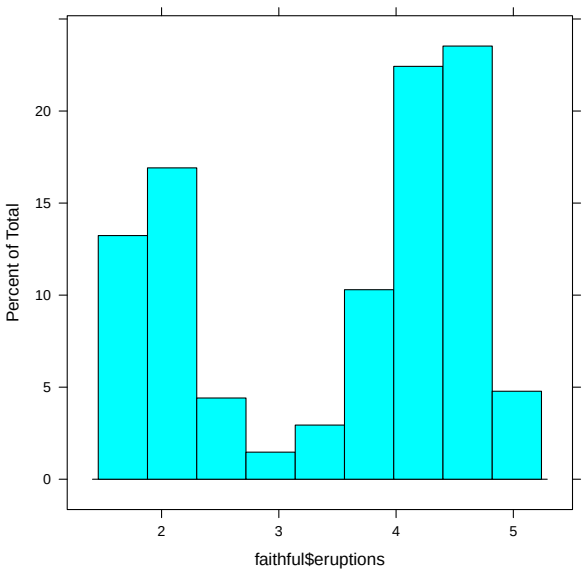


Figure 7: A more complex **grid** scene consisting of a simple plot produced by the `histogram()` function from the **lattice** package.

The `grid.ls()` function

For more complex scenes, the number of viewports and grobs can make it difficult to consume the listing from `grid.ls()` and, as viewports and grobs become nested to greater depths, simple indenting can be insufficient to convey the nesting clearly.

One solution is to specify a different formatting function via the `print` argument to the `grid.ls()` function. For example, the following code lists all grobs and viewports from Figure 7, but with only one line for each grob. The nesting of viewports is shown by listing the full viewport path to each grob. Figure 8 shows the resulting output.

```
> grid.ls(viewports=TRUE, print=grobPathListing)
```

Another solution is to capture (rather than just print) the result from `grid.ls()`. This is a list object containing a lot of information about the current scene and it can be processed computationally to answer more complex questions about the scene (see Figure 9).

```
> sceneListing <- grid.ls(viewports=TRUE,
+                         print=FALSE)

> do.call("cbind", sceneListing)
```

The `showGrob()` function

In a more complex scene, it is common for grobs to overlap each other, which can result in a messy labelling from the `showGrob()` function. Another problem is that text grobs do not label well because the labelling text is hard to read when overlaid on the text that is being labelled. One possible solution is to vary the graphical parameters used in the labelling. For example, the following code sets the fill colour for the grob bounding rectangles to be opaque (see Figure 10).

```
> showGrob(gp=gpar(fill=rgb(1, .85, .85)))
```

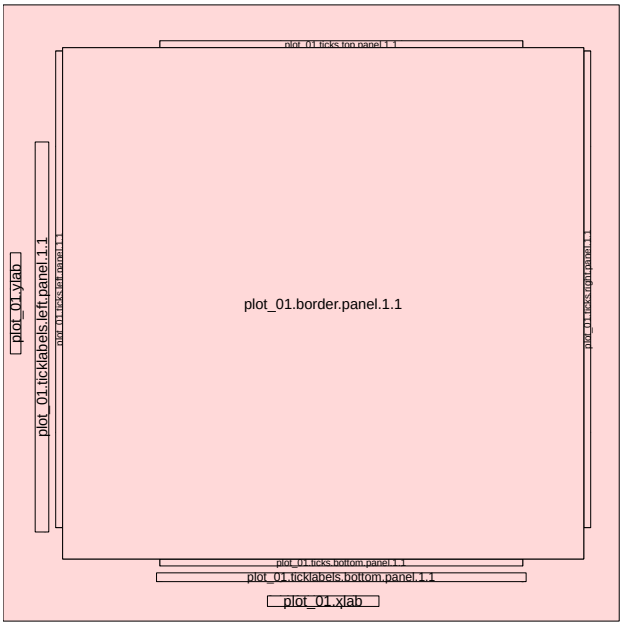


Figure 10: The **lattice** plot from Figure 7 with labelling added by the `showGrob()` function to show the locations and names of the grobs that were used to draw the scene.

One problem with this solution is that some overlapping grobs are not visible at all. To solve this, the `gPath` argument can be used to specify a particular grob to label. The following code uses this approach to label just the rectangle grob called `"plot_01.histogram.rect.panel.1.1"` (the rectangle grob that draws the histogram bars; see Figure 11).

```
> showGrob(gPath="plot_01.histogram.rect.panel.1.1")
```

The `showViewport()` function

In complex scenes, it is also very common for viewports to overlap each other. It is possible to display just a specific viewport with `showViewport()`, by supplying a viewport path as the first argument, but another option is to draw all viewports separately via the `leaves` argument. The following code demonstrates this approach and the result is shown in Figure 12.

In this case, there are eight viewports to display, so eight sub-regions have been drawn. Each sub-region represents an entire page, within which the location of one viewport is indicated with a shaded region. For example, the viewport `"plot_01."` takes up the entire page, but the viewport `"plot_01.xlab.vp"`

```
ROOT                                                    | plot_01.background
ROOT::plot_01.toplevel.vp::plot_01.xlab.vp              | plot_01.xlab
ROOT::plot_01.toplevel.vp::plot_01.ylab.vp              | plot_01.ylab
ROOT::plot_01.toplevel.vp::plot_01.strip.1.1.off.vp     | plot_01.ticks.top.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.strip.left.1.1.off.vp | plot_01.ticks.left.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.strip.left.1.1.off.vp | plot_01.ticklabels.left.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.off.vp     | plot_01.ticks.bottom.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.off.vp     | plot_01.ticklabels.bottom.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.off.vp     | plot_01.ticks.right.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.vp         | plot_01.histogram.baseline.lines.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.vp         | plot_01.histogram.rect.panel.1.1
ROOT::plot_01.toplevel.vp::plot_01.panel.1.1.off.vp     | plot_01.border.panel.1.1
```

Figure 8: A listing of the grobs and viewports from Figure 7 produced by `grid.ls()`.

```
                name gDepth vpDepth gPath                                   vpPath        type
1                ROOT      0       0                                                 vpListing
2   plot_01.background      0       1                                     ROOT grobListing
3  plot_01.toplevel.vp      0       1                                     ROOT   vpListing
4      plot_01.xlab.vp      0       2                    ROOT::plot_01.toplevel.vp   vpListing
5         plot_01.xlab      0       3 ROOT::plot_01.toplevel.vp::plot_01.xlab.vp grobListing
6                    1      0       3 ROOT::plot_01.toplevel.vp::plot_01.xlab.vp vpUpListing
7      plot_01.ylab.vp      0       2                    ROOT::plot_01.toplevel.vp   vpListing
8         plot_01.ylab      0       3 ROOT::plot_01.toplevel.vp::plot_01.ylab.vp grobListing
9                    1      0       3 ROOT::plot_01.toplevel.vp::plot_01.ylab.vp vpUpListing
10   plot_01.figure.vp      0       2                    ROOT::plot_01.toplevel.vp   vpListing
```

Figure 9: The raw result that is returned by a `grid.ls()` call for the scene in Figure 7. Only the first 10 lines of information is shown.

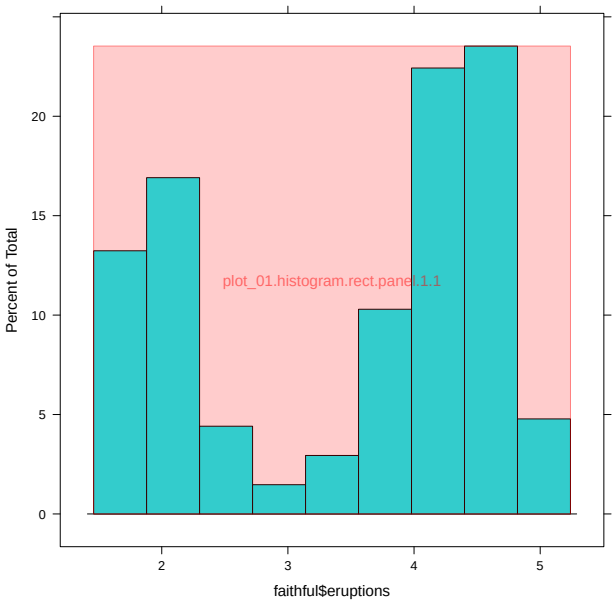


Figure 11: The **lattice** plot from Figure 7 with labelling added by the `showGrob()` function to show the location of grob `"plot_01.histogram.rect.panel.1.1"`.

only occupies a narrow strip towards the bottom of the page. Some viewports, for example, "plot_01.strip.1.1.off.vp", have zero height or zero width so appear as just straight lines.

```
> showViewport(newpage=TRUE, leaves=TRUE,
+              col="black")
```

Figure 12: The result of calling `showViewport()` to display the viewports used to draw the scene in Figure 7, with each viewport displayed on its own in a separate "mini page" to overcome the fact that several viewports overlap each other.

The `gridTree()` function

One advantage of the `gridTree()` function is that it is immune to the overlap of grobs and viewports in a scene. This is because this sort of display emphasizes the conceptual structure of the scene rather than reflecting the location of grobs and viewports on the page.

The following code produces a scene graph for the **lattice** plot from Figure 7 and the result is shown in Figure 13.

```
> gridTree()
```

One problem that does arise with the `gridTree()` function is that the grob and viewport names, which are used to label the nodes of the scene graph, can become too small to read.

The following code demonstrates this problem with an example plot from the **ggplot2** package. The plot is shown in Figure 14 and the scene graph generated by `gridTree()` is shown in Figure 15.

```
> library(ggplot2)
> qplot(faithful$eruptions, binwidth=.5)
```

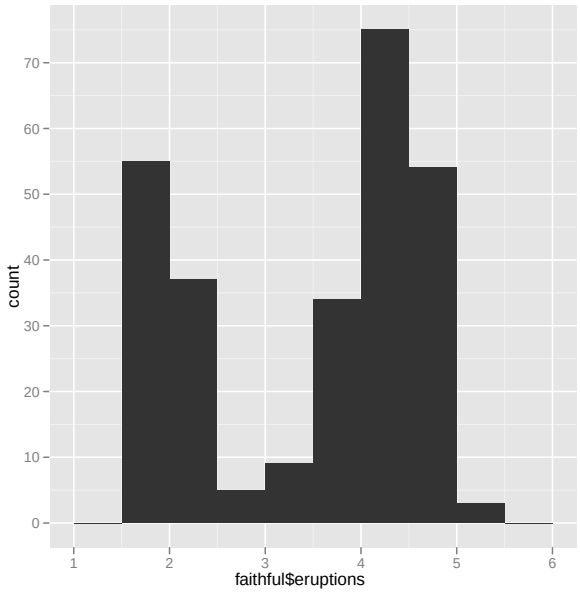


Figure 14: A more complex **grid** scene consisting of a simple plot produced by the `qplot()` function from the **ggplot2** package.

Although it is impossible to read the names of individual grobs and viewports on this graph, it is still interesting to compare the structure of this scene with the graph from the **lattice** plot in Figure 13. The graph clearly shows that the **lattice** package uses two levels of viewports, but only simple grobs, while the **ggplot2** package has a single, relatively complex, gTree that contains numerous other grobs, gTrees and viewports.

Interactive tools

The problem of unreadable labels on a scene graph may be alleviated by using the `gridTreeTips()` function, from the **gridDebug** package. This makes use of the **gridSVG** package (Murrell and Potter, 2011) to produce an SVG version of the scene graph with simple interaction added so that, when the mouse hovers over a node in the scene graph, a tooltip pops up to show the name of the node. Figure 16 shows an example of the output from this function (as viewed in Firefox).

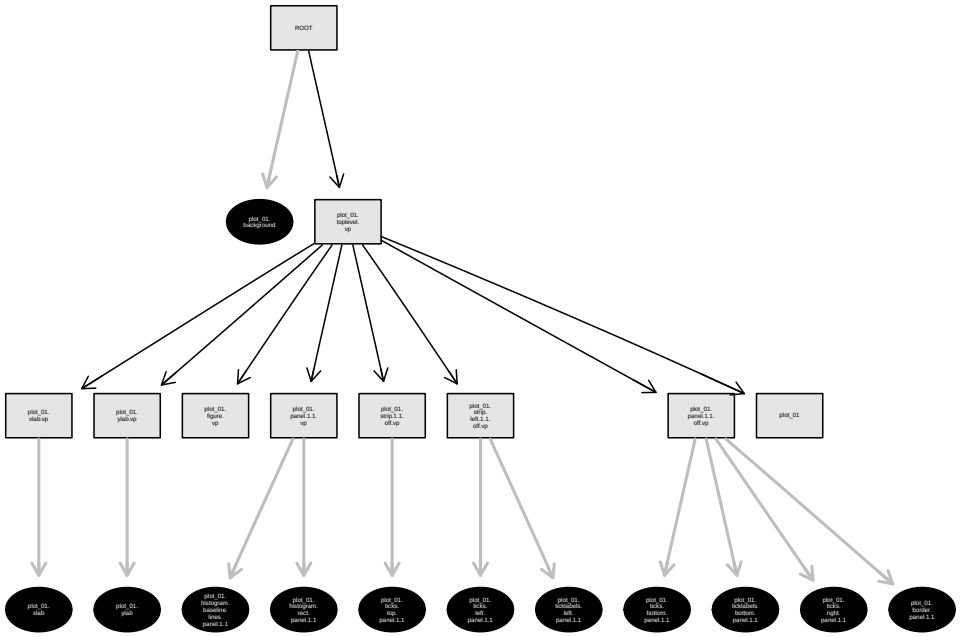


Figure 13: A node-and-edge graph of the scene from Figure 7

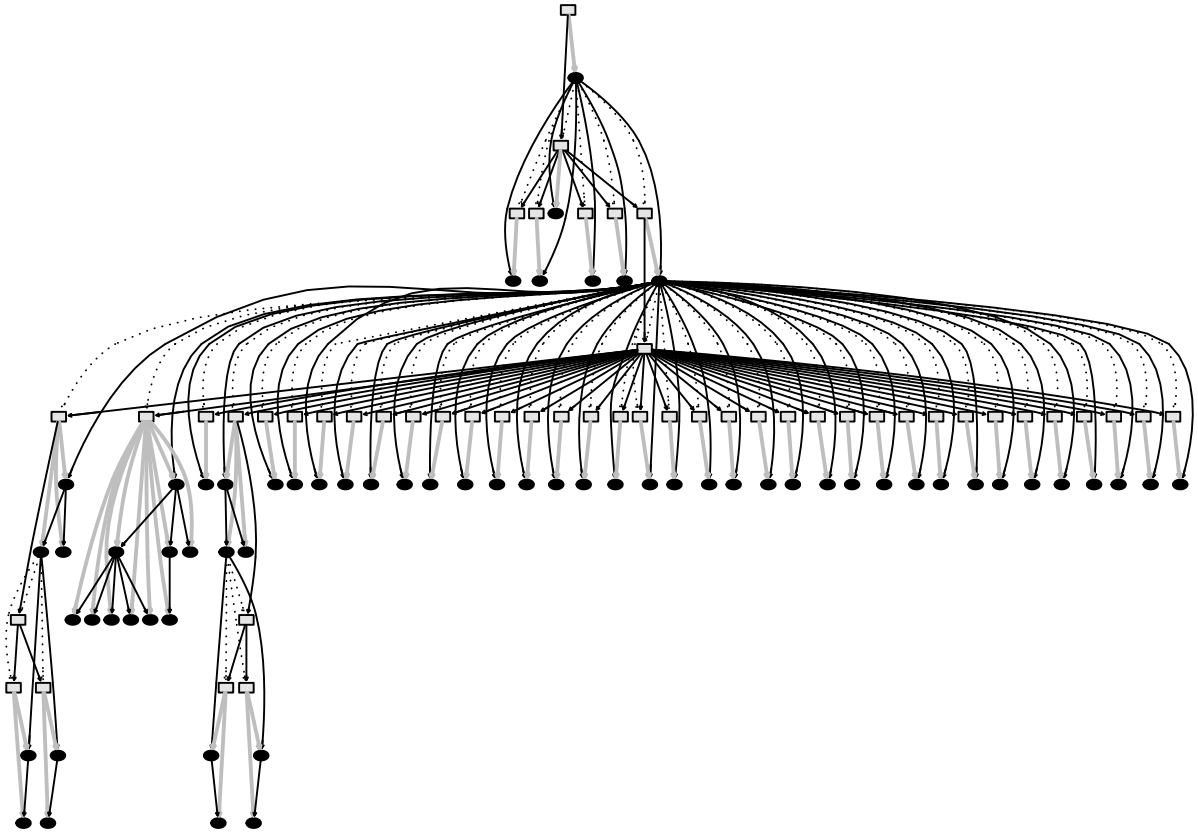


Figure 15: A node-and-edge graph of the scene from Figure 14. This scene graph was produced using **ggplot2** version 0.8.9. More recent versions of **ggplot2** will produce a different scene graph.

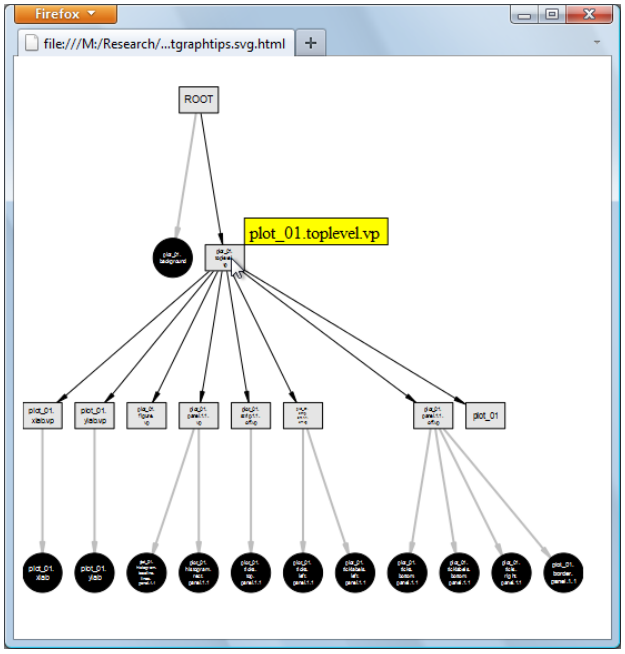


Figure 16: A node-and-edge graph of the scene from Figure 14 in SVG format so that, when the mouse hovers over a node on the graph, a tooltip shows the name of the node. The mouse is hovering over the node for the viewport called `"plot_01.toplevel.vp"`.

Another function from the **gridDebug** package, which also makes use of **gridSVG**, is the `grobBrowser()` function. This takes any **grid** scene and produces an SVG version of the scene that also contains tooltips. In this case, whenever the mouse hovers over a grob in the scene, a tooltip pops up to show the name of the grob. Figure 17 shows an example of the output from this function (as viewed in Firefox).

Tools in other packages

The **playwith** package (Andrews, 2010) also provides some tools for exploring the grobs in a **grid** scene. The `showGrobsBB()` function produces a similar result to `showGrob()` and `identifyGrob()` allows the user to click within a normal R graphics device to identify grobs. If the click occurs within the bounding box of a grob then the name of that grob is returned as the result. The result may be several grob names if there are overlapping grobs.

Conclusions

This article has described several tools that assist with the debugging of **grid** graphics code, whether that is trying to understand someone else's code, trying to understand your own code, or trying to explain **grid** code to someone else.

The tools provide various ways to view the names of grobs and viewports that were used to draw a scene, the relationships between the grobs and viewports, and where those grobs and viewports end up when drawn on the page.

Each of the tools has various weaknesses, so it may be necessary to use them in combination with each other in order to gain a complete understanding of a complex scene.

Acknowledgements

Many thanks to the anonymous reviewers for their useful comments and suggestions.

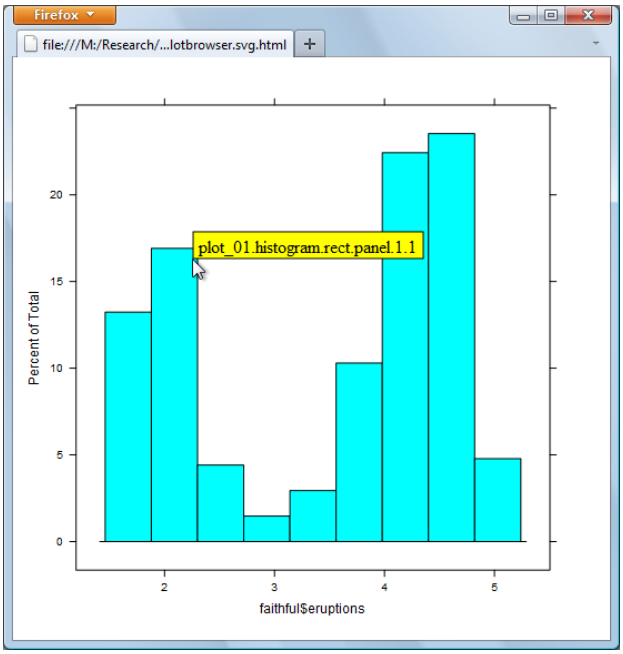


Figure 17: The scene from Figure 14 in SVG format so that, when the mouse hovers over a grob in the scene, a tooltip shows the name of the grob. The mouse is hovering over one of the bars in the histogram, which corresponds to the grob called `"plot_01.histogram.rect.panel.1.1"`.

Bibliography

F. Andrews. *playwith: A GUI for interactive plots using GTK+*, 2010. URL `http://CRAN.R-project.org/package=playwith`. R package version 0.9-53. [p26]

R. Gentleman, E. Whalen, W. Huber, and S. Falcon. *graph: A package to handle graph data structures*, 2010. URL `http://CRAN.R-project.org/package=graph`. R package version 1.28.0. [p21]

J. Gentry, L. Long, R. Gentleman, S. Falcon, F. Hahne, D. Sarkar, and K. Hansen. *Rgraphviz: Provides plot-*

ting capabilities for R graph objects, 2010. R package version 1.23.6. [p21]

P. Murrell. *gridGraphviz: Drawing Node-and-Edge Graphs Using Grid*, 2011. R package version 0.1. [p21]

P. Murrell and V. Ly. *gridDebug: Debugging Grid Graphics*, 2011. R package version 0.2. [p20]

P. Murrell and S. Potter. *gridSVG: Export grid graphics as SVG*, 2011. R package version 0.7-0. [p24]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p19]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book. [p19]

Paul Murrell
Department of Statistics
The University of Auckland
New Zealand
paul@stat.auckland.ac.nz

Velvet Ly
Department of Statistics
The University of Auckland
New Zealand

frailtyHL: A Package for Fitting Frailty Models with H-likelihood

by Il Do Ha, Maengseok Noh and Youngjo Lee

Abstract We present the **frailtyHL** package for fitting semi-parametric frailty models using h-likelihood. This package allows lognormal or gamma frailties for random-effect distribution, and it fits shared or multilevel frailty models for correlated survival data. Functions are provided to format and summarize the **frailtyHL** results. The estimates of fixed effects and frailty parameters and their standard errors are calculated. We illustrate the use of our package with three well-known data sets and compare our results with various alternative R-procedures.

Introduction

Frailty models with a non-parametric baseline hazard have been widely adopted for the analysis of survival data (Hougaard, 2000; Duchateau and Janssen, 2008). The **frailtyHL** package (Ha et al., 2012) implements h-likelihood procedures (Ha et al., 2001; Ha and Lee, 2003, 2005) for frailty models. The package fits Cox proportional hazards models (PHMs) with random effects, or frailties. The lognormal or gamma distribution can be adopted as the frailty distribution, corresponding to the normal or log-gamma distribution for the log frailties. H-likelihood obviates the need for marginalization over the frailty distribution, thereby providing a statistically efficient procedure for various random-effect models (Lee et al., 2006).

The main function is `frailtyHL()`. For instance,

```
> frailtyHL(Surv(time, status) ~ x + (1|id),
+           RandDist = "Normal",
+           mord = 0, dord = 1,
+           Maxiter = 200, convergence = 10^-6,
+           varfixed = FALSE, varinit = 0.1)
```

fits a lognormal frailty model. The first argument is a formula object, with the response on the left of a `~` operator, and the terms for the fixed and random effects on the right. The response is a survival object as returned by the `Surv` function (Therneau, 2011). Here, `time` and `status` denote survival time and censoring indicator taking value 1 or 0 for uncensored or censored observations, respectively; `x` denotes a fixed covariate and `id` denotes the subject identifier. The expression `(1|id)` specifies a random intercept model (`(x|id)` would specify a random intercept, random slope model). The parameters `mord` and `dord` are the orders of Laplace approximations to fit the mean parameters (`mord` = 0 or 1) and the dispersion parameters (`dord` = 1 or 2), respectively. The `Maxiter` parameter specifies the maximum number of iterations and `convergence` specifies the tolerance of the convergence criterion. If `varfixed` is specified as `TRUE` (or `FALSE`), the value of one or more of the variance terms for the frailties is fixed (or estimated) with starting value (e.g. 0.1) given by `varinit`.

Previously, frailty models have been implemented in several R functions such as the `coxph()` function in the **survival** package (Therneau, 2010) and the `coxme()` function in the **coxme** package (Therneau, 2011), based on penalized partial likelihood (PPL), the `phmm()` function in the **phmm** package (Donohue and Xu, 2012), based on a Monte Carlo EM (MCEM) method, and the `frailtyPenal()` function in the **frailtypack** package (Gonzalez et al., 2012), based on penalized marginal likelihood. The **phmm** package fits one-component frailty models, although it does allow multivariate frailties. The `coxme()` function can fit the multi-component model as shown in Example 2. Results from **frailtyHL** are compared with those from **survival**, **coxme** and **phmm**.

Recently, the h-likelihood procedures of Lee and Nelder (1996) for fitting hierarchical generalized linear models (HGLMs) have been implemented using the `hglm()` function in the **hglm** package (Alam et al., 2010), the `HGLMfit()` function in the **HGLMMM** package (Molas, 2010) and the `dhglmfit()` function in the **dhglm** package (Noh and Lee, 2011). The **frailtyHL** package is for fitting frailty models with a non-parametric baseline hazard, providing estimates of fixed effects, random effects, and variance components as well as their standard errors. In addition, it provides a statistical test for the variance components of frailties and also three AIC criteria for the model selection.

Frailty models

The **frailtyHL** package makes it possible to

1. fit models with log-normal and gamma frailty distributions and
2. estimate variance components when the frailty structure is shared or nested.

For illustration, we present two models below and show how to fit these models using `frailtyHL()` in the **Examples** section.

One-component frailty models

Suppose that data consist of right censored time-to-event observations from q subjects (or clusters), with

n_i observations each; $i = 1,\ldots,q$. Let T_{ij} be the survival time for the jth observation of the ith subject; $j = 1,\ldots,n_i$. Here, $n = \sum_i n_i$ is the total sample size and n_i is the cluster size. Let C_{ij} be the corresponding censoring time and let $y_{ij} = \min\{T_{ij}, C_{ij}\}$ and $\delta_{ij} = I(T_{ij} \le C_{ij})$ be the observable random variables, where $I(\cdot)$ is the indicator function.

Given the common unobserved frailty for the ith subject u_i, the conditional hazard function of T_{ij} is of the form

$$\lambda_{ij}(t|u_i) = \lambda_0(t)\exp(x_{ij}^T\beta)u_i, \qquad (1)$$

where $\lambda_0(\cdot)$ is an unspecified baseline hazard function and $\beta = (\beta_1,\ldots,\beta_p)^T$ is a vector of regression parameters for the fixed covariates x_{ij}. Here, the term $x_{ij}^T\beta$ does not include an intercept term because of identifiability. Assume that the frailties u_i are independent and identically distributed (i.i.d) random variables with a frailty parameter α; the gamma and lognormal frailty models assume gamma and lognormal distributions for u_i, respectively. That is, our package allows

- gamma frailty with $E(u_i) = 1$ and $\text{var}(u_i) = \alpha$, and
- lognormal frailty having $v_i = \log u_i \sim N(0,\alpha)$.

The model represented by (1) is known as a shared or one-component frailty model (Hougaard, 2000).

Multi-component frailty models

We can fit a multi-component model as below (Ha et al., 2007):

$$X\beta + Z^{(1)}v^{(1)} + Z^{(2)}v^{(2)} + \cdots + Z^{(k)}v^{(k)}, \qquad (2)$$

X is the $n \times p$ model matrix, $Z^{(r)}$ $(r = 1,2,\ldots,k)$ are the $n \times q_r$ model matrices corresponding to the $q_r \times 1$ frailties $v^{(r)}$, and $v^{(r)}$ and $v^{(l)}$ are independent for $r \ne l$.

For example, the CGD data (Fleming and Harrington, 1991) have a multilevel structure in which patients nested within centers have recurrent infection times.

Later, we analyze these data using model (2) with $k = 2$. Here, the frailty structures are:

$$v^{(1)}\text{: center frailty} \sim N(0,\alpha_1 A_1),$$
$$v^{(2)}\text{: patient frailty} \sim N(0,\alpha_2 A_2),$$

where $A_r = I_r$ $(r = 1,2)$ are the $q_r \times q_r$ identity matrices, and q_1 and q_2 are the number of centers and patients, respectively. Notice that the corresponding $Z^{(1)}$ to $v^{(1)}$ (or $Z^{(2)}$ to $v^{(2)}$) is, respectively, a matrix of indicator variables such that $Z^{(1)}_{st} = 1$ (or $Z^{(2)}_{st} = 1$) if observation s is a member of center (or patient) t and 0 otherwise (Therneau and Grambsch, 2000). This is called the multilevel frailty model (Yau, 2001; Ha et al., 2007).

H-likelihood theory

The h-likelihood h (Ha et al., 2001) for frailty model (1) is defined by

$$h = h(\beta,\lambda_0,\alpha) = \ell_0 + \ell_1, \qquad (3)$$

where

$$\begin{aligned}\ell_0 &= \sum_{ij}\log f(y_{ij},\delta_{ij}|u_i;\beta,\lambda_0)\\ &= \sum_{ij}\delta_{ij}\{\log\lambda_0(y_{ij}) + \eta_{ij}\} - \sum_{ij}\Lambda_0(y_{ij})\exp(\eta_{ij})\end{aligned}$$

is the sum of conditional log densities for y_{ij} and δ_{ij} given u_i, and

$$\ell_1 = \sum_i \log f(v_i;\alpha)$$

is the sum of log densities for $v_i = \log u_i$ with parameter α. Here, $\eta_{ij} = x_{ij}^T\beta + v_i$ is the linear predictor for the hazards, and

$$\Lambda_0(t) = \int_0^t \lambda_0(k)dk$$

is the baseline cumulative hazard function.

The functional form of $\lambda_0(t)$ in (1) is unknown; following Breslow (1972), we consider $\Lambda_0(t)$ to be a step function with jumps at the observed event times:

$$\Lambda_0(t) = \sum_{k:y_{(k)}\le t}\lambda_{0k}$$

where $y_{(k)}$ is the kth $(k = 1,\ldots,l)$ smallest distinct event time among the y_{ij}'s, and $\lambda_{0k} = \lambda_0(y_{(k)})$. However, the dimension of $\lambda_0 = (\lambda_{01},\ldots,\lambda_{0l})^T$ increases with the sample size n. For inference, Ha et al. (2001) proposed the use of the profile h-likelihood with λ_0 eliminated, $h^* \equiv h|_{\lambda_0=\widehat{\lambda}_0}$, given by

$$h^* = h^*(\beta,\alpha) = \ell_0^* + \ell_1, \qquad (4)$$

where

$$\begin{aligned}\ell_0^* &= \sum_{ij}\log f^*(y_{ij},\delta_{ij}|u_i;\beta)\\ &= \sum_{ij} f(y_{ij},\delta_{ij}|u_i;\beta,\widehat{\lambda}_0)\end{aligned}$$

does not depend on λ_0, and

$$\widehat{\lambda}_{0k}(\beta,v) = \frac{d_{(k)}}{\sum_{(i,j)\in R_{(k)}}\exp(\eta_{ij})},$$

are solutions of the estimating equations, $\partial h/\partial\lambda_{0k} = 0$, for $k = 1,\ldots,l$. Here, $d_{(k)}$ is the number of events at $y_{(k)}$ and

$$R_{(k)} = R(y_{(k)}) = \{(i,j) : y_{ij} \ge y_{(k)}\}$$

is the risk set at $y_{(k)}$. Therneau and Grambsch (2000) and Ripatti and Palmgren (2000) proposed an h-likelihood (3), while using the partial likelihood (Breslow, 1974) for ℓ_0. They call it the penalized partial likelihood (PPL) h_p, defined by

$$h_p(\beta, v, \alpha) = \sum_{ij} \delta_{ij}\eta_{ij} - \sum_k d_{(k)} \log\left\{ \sum_{ij \in R_{(k)}} \exp(\eta_{ij}) \right\} + \ell_1.$$

Furthermore, Ha et al. (2001) and Ha et al. (2010) have shown that h^* is proportional to the PPL h_p because

$$\begin{aligned} h^* &= \sum_k d_{(k)} \log \widehat{\lambda}_{0k} + \sum_{ij} \delta_{ij}\eta_{ij} - \sum_k d_{(k)} \; + \; \ell_1 \\ &= h_p + \sum_k d_{(k)}\{\log d_{(k)} - 1\}, \end{aligned}$$

where $\sum_k d_{(k)}\{\log d_{(k)} - 1\}$ is a constant that does not depend upon unknown parameters. Notice here that PPL h_p does not depend on nuisance parameters λ_0. Thus, Lee and Nelder's (1996; 2001) h-likelihood procedure for HGLMs can be directly used to fit frailty models based on h_p (Ha et al., 2010).

Review of estimation procedures

Lee and Nelder (1996, 2001) have proposed the use of the Laplace approximation based on the h-likelihood when the marginal likelihood, $m = \log\{\int \exp(h) dv\}$, is hard to obtain. To reduce the bias in small cluster sizes higher-order approximations for the mean (β) and the frailty variance (α) have been developed. The lower-order approximation is computationally efficient, but could have large biases when cluster sizes are small (Ha and Lee, 2003, 2005; Ha et al., 2010).

The h-likelihood procedures use the Breslow method for handling tied event times, while the PPL procedures allow the Efron method. For estimating β, the h-likelihood methods allow the Laplace approximation $p_v(h_p)$ to a marginal partial likelihood $m_p = \log\{\int \exp(h_p) dv\}$, but the PPL procedures do not. For estimating α, the PPL methods use adjusted profile h-likelihoods $p_v(h_p)$ and $p_{\beta,v}(h_p)$ which give maximum partial likelihood (MPL) and restricted maximum partial likelihood (REMPL) estimators, respectively. In contrast, the h-likelihood method uses the restricted partial likelihood (based upon the first-order Laplace approximation $p_{\beta,v}(h_p)$ or the second-order Laplace approximation $s_{\beta,v}(h_p)$) for REMPL estimators. Here $p_v(h_p)$ and $p_{\beta,v}(h_p)$ are defined as follows:

$$p_v(h_p) = [h_p - \frac{1}{2}\log\det\{D(h_p, v)/(2\pi)\}]|_{v=\hat{v}},$$

where $D(h_p, v) = -\partial^2 h_p/\partial v^2$ and $\hat{v}$ solves $\partial h_p/\partial v = 0$, which is the first-order Laplace approximation of m_p, and

$$p_{\beta,v}(h_p) = \left[h_p - \frac{1}{2}\log\det\left\{\frac{D(h_p, (\beta, v))}{2\pi}\right\}\right]\Big|_{\beta=\hat{\beta}, v=\hat{v}},$$

where $D(h_p, (\beta, v)) = -\partial^2 h_p/\partial(\beta, v)^2$ and $(\hat{\beta}, \hat{v})$ solves $\partial h_p/\partial(\beta, v) = 0$, which becomes the Cox and Reid (1987) adjusted profile marginal likelihood, eliminating fixed effects β by conditioning their asymptotic sufficient statistics $\hat{\beta}$, in addition to eliminating random effects v by the first-order Laplace approximation (Ha and Lee, 2003; Lee et al., 2006). The corresponding second-order approximation is

$$s_{\beta,v}(h_p) = p_{\beta,v}(h_p) - \{F(h_p)/24\},$$

with

$$F(h_p) = \mathrm{tr}\left[-\left\{3\frac{\partial^4 h_p}{\partial v^4} + 5\frac{\partial^3 h_p}{\partial v^3} D(h_p, v)^{-1} \frac{\partial^3 h_p}{\partial v^3}\right\} \times D(h_p, v)^{-2}\right]\Big|_{v=\widehat{v}}.$$

To reduce the computational burden Ha et al. (2010) used $F(h)$ instead of $F(h_p)$.

Table 1: Estimation criteria for h-likelihood (HL(`mord`, `dord`)), PPL (`coxph()`, `coxme()`) and marginal likelihood (`phmm()`) for lognormal (LN) and gamma frailty models (FM)

Method	Criterion β	Criterion α	Literature
HL(0,1)	h_p	$p_{\beta,v}(h_p)$	Ha and Lee (2003)
HL(0,2)	h_p	$s_{\beta,v}(h_p)$	Ha and Lee (2003)
HL(1,1)	$p_v(h_p)$	$p_{\beta,v}(h_p)$	Ha et al. (2012)
HL(1,2)	$p_v(h_p)$	$s_{\beta,v}(h_p)$	Ha et al. (2012)
`coxph()`	h_p	$p_{\beta,v}(h_p)$	Therneau (2010) for LN FM
`coxph()`	h_p	m	Therneau (2010) for gamma FM
`coxme()`	h_p	$p_v(h_p)$	Therneau (2011) for LN FM
`phmm()`	m	m	Donohue and Xu (2012) for LN FM

Table 1 shows historical developments of estimating criteria for frailty models. The `frailtyHL()` function provides estimators based on the h-likelihood. As the orders in `mord` and `dord` increase, the bias of estimator is reduced, but the calculation becomes computationally intensive due to the extra terms.

We recommend the use of HL(1,1) for the lognormal frailty and of HL(1,2) for the gamma frailty. However, HL(0,1) for the lognormal frailty and HL(0,2) for the gamma frailty often perform well if α is not large. Note that the variance matrices of $\hat{\tau} = (\hat{\beta}, \hat{v})$ and $\hat{\alpha}$ are directly obtained from the Hessian matrices, $\{-\partial^2 h_p/\partial\tau^2\}^{-1}$ and $\{-\partial^2 p_{\beta,v}(h_p)/\partial\alpha^2\}^{-1}$, respectively; the **frailtyHL** package provides the standard errors (SEs) of $\hat{\alpha}$ as well as $\hat{\beta}$. For the use of standard errors of $\hat{v} - v$, see Lee and Ha (2010), Lee et al (2011) and Ha et al. (2011).

Based on the PPL methods, the `coxph()` and `coxme()` functions, respectively, implement REMPL and MPL estimators for α in the lognormal frailty model, and the `coxph()` function the maximum likelihood (ML) estimators, maximizing the marginal likelihood m, for α in the gamma frailty model. For comparison, we present the Breslow and Efron methods for handling ties in survival times in the `coxph()` and `coxme()` functions; Therneau (2010) recommended the Efron method. For the lognormal frailty the ML estimator maximizing m is available via the `phmm()` function, but care must be taken to ensure that the MCEM algorithm has converged (Donohue and Xu, 2012). However, the ML estimator can be biased when the number of nuisance parameters increases with the sample size (Ha et al., 2010).

Furthermore, for the lognormal frailty the `coxph()` function uses the existing codes in linear mixed models so that it misses the $\partial\hat{v}/\partial\alpha$ term in solving the score equation $\partial p_{\beta,v}(h_p)/\partial\alpha = 0$; this can lead to an underestimation of the parameters, especially when the number of subjects q is large or censoring is high (Lee et al., 2006; Ha et al., 2010). To overcome this problem, in gamma frailty models Therneau and Grambsch (2000) develop the code for the ML estimator for α.

Fitting algorithm

Suppose that HL(0,1) is used. The fitting algorithm is as follows:

Step 1: Take $(0, 0, 0.1)$ as initial estimates of components of (β, v, α).

Step 2: Given $\hat{\alpha}$, new estimates $(\hat{\beta}, \hat{v})$ are obtained by solving the joint estimating equations $\partial h_p/\partial(\beta, v) = \{\partial h/\partial(\beta, v)\}|_{\lambda_0=\hat{\lambda}_0} = 0$; then, given $(\hat{\beta}, \hat{v})$, new estimates $\hat{\alpha}$ are obtained by solving $\partial p_{\beta,v}(h_p)/\partial\alpha = 0$.

Step 3: Repeat Step 2 until the maximum absolute difference between the previous and current estimates for (β, v) and α is less than 10^{-6}.

After convergence, we compute the estimates of the standard errors of $\hat{\beta}$ and $\hat{\alpha}$.

Illustration: kidney infection data

To demonstrate differences of various estimation methods in small cluster size $n_i \equiv 2$, we use the kidney infection data (McGilchrist and Aisbett, 1991). The data consist of times until the first and second recurrences ($n_i \equiv 2$) of kidney infection in 38 ($q = 38$) patients using a portable dialysis machine. Each survival time (`time`) is the time until infection since the insertion of the catheter. The survival times for the same patient are likely to be related because of a shared frailty describing the common patient's effect. The catheter is later removed if infection occurs and can be removed for other reasons, which we regard as censoring; about 24% of the data were censored.

We fit frailty models with two covariates, the sex (1 = male; 2 = female) and age of each patient, using the functions (`frailtyHL()`, `coxph()` , `coxme()` and `phmm()`) in the four packages. The results are summarized in Table 2.

Table 2: Comparison of different estimation methods for the kidney infection data

Method	Sex $\hat{\beta}_1$ (SE)	Age $\hat{\beta}_2$ (SE)	Patient $\hat{\alpha}$ (SE)
lognormal model			
HL(0,1)	-1.380	0.005	0.535
	(0.431)	(0.012)	(0.338)
HL(1,1)	-1.414	0.005	0.545
	(0.432)	(0.012)	(0.340)
`coxph()`	-1.388	0.005	0.551
(Breslow)	(0.441)	(0.012)	(–)
`coxph()`	-1.411	0.005	0.569
(Efron)	(0.445)	(0.013)	(–)
`coxme()`	-1.332	0.005	0.440
(Breslow)	(0.414)	(0.012)	(–)
`coxme()`	-1.355	0.004	0.456
(Efron)	(0.417)	(0.012)	(–)
`phmm()`	-1.329	0.004	0.378
	(0.452)	(0.012)	(–)
gamma model			
HL(0,2)	-1.691	0.007	0.561
	(0.483)	(0.013)	(0.280)
HL(1,2)	-1.730	0.007	0.570
	(0.485)	(0.013)	(0.281)
`coxph()`	-1.557	0.005	0.398
(Breslow)	(0.456)	(0.012)	(–)
`coxph()`	-1.587	0.005	0.412
(Efron)	(0.461)	(0.012)	(–)

In PPL procedures (`coxph()` and `coxme()`), the Breslow method provides slightly smaller estimate for α than the Efron method. In the lognormal frailty, REMPL procedures (`frailtyHL()` and `coxph()`) give larger estimates for α than ML (`phmm()`) and MPL (`coxme()`) procedures. However, both ML and MPL estimates from `phmm()` and `coxme()` are somewhat

different when cluster size is small, $n_i \equiv 2$ for all i. For the gamma frailty, `coxph()` uses the ML procedure, but it still gives smaller estimate for α than the REMPL (h-likelihood) procedures. Compared with the h-likelihood methods, PPL methods are computationally more efficient, but could have larger biases (Ha et al., 2010).

Potential future developments

The current version of the **frailtyHL** package allows multi-component (multilevel) frailties. Allowance for correlation (Ha et al., 2011) between random effects, for example correlations between random center effect and random treatment effect, is currently in progress. Other developments include dispersion frailty models based on double HGLMs (Lee and Nelder, 2006), which allow fixed and random effects in dispersion parts (i.e. variance of random effects) of the model as well as the hazard functions, and joint modelling (Ha et al, 2003) of HGLMs and frailty models.

Examples

Example 1: Lognormal and gamma frailty models on rat data

The data set presented by Mantel et al. (1977) is based on a tumorigenesis study of 50 ($q = 50$) litters of female rats. For each litter, one rat was selected to receive the drug and the other two rats were placebo-treated controls ($n_i \equiv 3$). Here each litter is treated as a cluster. The survival time (`time`) is the time to development of tumor, measured in weeks. Death before occurrence of tumor yields a right-censored observation; forty rats developed a tumor, leading to censoring of about 73%. The survival times for rats in a given litter may be correlated due to a random effect representing shared genetic or environmental effects.

We fit frailty models with one covariate, rx (1 = drug; 0 = placebo), using `frailtyHL()`. Below, we present the code and results for the lognormal frailty model with HL(1,1). The output from the R code shows that the effect of rx is significant (t-value = 2.823 with p-value = 0.005). That is, the rx group has a significantly higher risk than in control group. Here, the variance estimate of the frailty is $\hat{\alpha} = 0.427$ (with SE = 0.423).

Note that although we report the SE of α, one should not use it for testing the absence of frailty $\alpha = 0$ (Vaida and Xu, 2000). Such a null hypothesis is on the boundary of the parameter space, so that the critical value of an asymptotic $(\chi_0^2 + \chi_1^2)/2$ distribution is 2.71 at 5% significant level (Lee et al., 2006; Ha et al., 2011). The difference in deviance (based on REMPL) $-2p_{\beta,v}(h_p)$ between the Cox model without frailty and the lognormal frailty model is $364.15 - 362.56 = 1.59 (< 2.71)$, indicating that the frailty effect is non-significant, i.e. $\alpha = 0$. Here, the results from fitting the Cox model without frailty are available by adding the two arguments `varfixed = TRUE` and `varinit = 0` in the `frailtyHL()`: see below.

```
> library(survival)
> data(rats)
> frailtyHL(Surv(time,status) ~ rx + (1|litter),
+           data = rats,
+           varfixed = TRUE, varinit = 0)

iteration :
          4
convergence :
  4.801639e-09
[1] "converged"
[1] "Results from the Cox model"
[1] "Number of data : "
[1] 150
[1] "Number of event : "
[1] 40
[1] "Model for conditional hazard : "
Surv(time, status) ~ rx + (1 | litter)
[1] "Method : HL(0,1)"
[1] "Estimates from the mean model"
   Estimate Std. Error t-value  p-value
rx   0.8982     0.3174    2.83 0.004655
[1] "Estimates from the dispersion model"
       Estimate Std. Error
litter "0"      "NULL"
       -2h0  -2*hp -2*p_b,v(hp)
[1,] 363.69 363.69   364.15
       cAIC   mAIC   rAIC
[1,] 365.69 365.69 364.15



> frailtyHL(Surv(time,status) ~ rx + (1|litter),
+           data = rats, RandDist = "Normal",
+           mord = 1, dord = 1)

iteration :
          87
convergence :
   9.97616e-07
[1] "converged"
[1] "Results from the log-normal frailty model"
[1] "Number of data : "
[1] 150
[1] "Number of event : "
[1] 40
[1] "Model for conditional hazard : "
Surv(time, status) ~ rx + (1 | litter)
[1] "Method : HL(1,1)"
[1] "Estimates from the mean model"
   Estimate Std. Error t-value  p-value
rx   0.9107     0.3226   2.823 0.004754
[1] "Estimates from the dispersion model"
       Estimate Std. Error
litter   0.4272     0.4232
       -2h0  -2*hp -2*p_v(hp) -2*p_b,v(hp)
```

```
[1,] 335.97 397.36     362.14    362.56
       cAIC   mAIC   rAIC
[1,] 362.22 366.14 364.56
```

The code and results for the gamma frailty model with HL(1,2) are presented below. The output shows that these results are similar to those of the lognormal frailty, particularly for estimation of β. The deviance difference (based on REMPL) between the Cox model and gamma frailty model using the second-order approximation $-2s_{\beta,v}(h_p)$ is $364.15 - 362.12 = 2.03 (< 2.71)$, again indicating the absence of frailty effect (i.e. $\alpha = 0$) as evident in the lognormal frailty model.

```
> frailtyHL(Surv(time,status) ~ rx + (1|litter),
+           data = rats, RandDist = "Gamma",
+           mord = 1, dord = 2)

iteration :
         170
convergence :
  9.567765e-07
[1] "converged"
[1] "Results from the gamma frailty model"
[1] "Number of data : "
[1] 150
[1] "Number of event : "
[1] 40
[1] "Model for conditional hazard : "
Surv(time, status) ~ rx + (1 | litter)
[1] "Method : HL(1,2)"
[1] "Estimates from the mean model"
   Estimate Std. Error t-value  p-value
rx   0.9126     0.3236    2.82 0.004806
[1] "Estimates from the dispersion model"
       Estimate Std. Error
litter   0.5757     0.5977
     -2h0  -2*hp  -2*p_v(hp) -2*s_v(hp)
[1,] 331.60 413.85   365.35   361.71
     -2*p_b,v(hp) -2*s_b,v(hp)
      365.77      362.12
      cAIC   mAIC   rAIC
[1,] 365.30 365.71 364.12
```

For the selection of a model among nested or non-nested models such as lognormal and gamma frailty models, we may use three Akaike information criteria (AIC) (Lee et al., 2006; Donohue et al., 2011; Ha et al., 2007) based on conditional likelihood, marginal likelihood and restricted likelihood, respectively, defined by

$$\begin{aligned} \text{cAIC} &= -2h_0 + 2\text{df}_\text{c}, \\ \text{mAIC} &= -2p_v(h_p) + 2\text{df}_\text{m}, \\ \text{rAIC} &= -2p_{\beta,v}(h_p) + 2\text{df}_\text{r}, \end{aligned}$$

where $h_0 = \ell_0^*$ in (4), and

$$\text{df}_\text{c} = \text{trace}\left\{ D^{-1}(h_p,(\beta,v)) D(h_0,(\beta,v)) \right\}$$

is an 'effective degrees of freedom adjustment' for estimating the fixed and random effects, computed using the Hessian matrices $D(h_p,(\beta,v)) = -\partial^2 h_p/\partial(\beta,v)^2$ and $D(h_0,(\beta,v)) = -\partial^2 h_0/\partial(\beta,v)^2$. Note here that df_m is the number of fixed parameters and df_r is the number of dispersion parameters (Ha et al., 2007). For calculation of the mAIC and rAIC of gamma frailty model using HL(0,2) or HL(1,2), we use the corresponding second-order approximations, defined by $\text{mAIC} = -2s_v(h_p) + 2\text{df}_\text{m}$ and $\text{rAIC} = -2s_{\beta,v}(h_p) + 2\text{df}_\text{r}$. We select a model to minimize the AIC values among models. If the AIC difference is larger than 1 the choice can be made (Sakamoto et al., 1986). However, if the difference is less than 1 a simpler model can be selected by a parsimony principle (Donohue et al., 2011).

In the data set, in the Cox model cAIC=365.69, mAIC=365.69 and rAIC=364.15, and in the lognormal frailty model cAIC=362.22, mAIC=366.14 and rAIC=364.56, and in the gamma frailty model cAIC=365.30, mAIC=365.71 and rAIC=364.12. The likelihood tests based upon the REMPL showed the absence of frailty effect ($\alpha = 0$), so that mAIC and rAIC of all the three models are similar. Thus, we may choose the parsimonious Cox model. However, the cAIC selects the lognormal model, indicating that this model could give better prediction.

Example 2: Multilevel frailty models on CGD infection data

The CGD data set presented by Fleming and Harrington (1991) consists of a placebo-controlled randomized trial of gamma interferon (rIFN-g) in the treatment of chronic granulomatous disease (CGD). 128 patients (`id`) from 13 centers ($q_1 = 13, q_2 = 128$) were tracked for around 1 year. The number (i.e. cluster size) of patients per center ranged from 4 to 26. The survival times (`tstop-tstart`) are the recurrent infection times of each patient from the different centers. Censoring occurred at the last observation for all patients, except one, who experienced a serious infection on the day he left the study; in the CGD study about 63% of the data were censored. The recurrent infection times for a given patient are likely to be correlated. However, each patient belongs to one of the 13 centers; hence, the correlation may also be attributed to a center effect.

Ignoring important random components may render invalid many of the traditional statistical analysis techniques. We fit a multilevel lognormal frailty model with two frailties and a single covariate, treat (rIFN-g, placebo), using `frailtyHL()`. Here, the two frailties are random center and patient effects. The code and results using HL(1,1) are provided below. The output shows that the effect of treatment is significant (t-value = -3.476 with p-value < 0.001), indicating that rIFN-g significantly reduces the rate of serious infection in CGD patients. The estimate of

variance of patient frailty ($\hat{\alpha}_2 = 1.002$) is considerably larger than variance of center frailty ($\hat{\alpha}_1 = 0.030$), indicating that the random-patient effect is more heterogeneous.

```
> library(survival)
> data(cgd)
> frailtyHL(Surv(tstop-tstart,status) ~ treat +
+             (1|center) + (1|id),
+           data = cgd,
+           RandDist = "Normal", mord = 1, dord = 1)

iteration :
         157
convergence :
  9.336249e-07
[1] "converged"
[1] "Results from the log-normal frailty model"
[1] "Number of data : "
[1] 203
[1] "Number of event : "
[1] 76
[1] "Model for conditional hazard : "
Surv(tstop-tstart,status)~treat+(1|center)+(1|id)
[1] "Method : HL(1,1)"
[1] "Estimates from the mean model"
            Estimate Std. Error t-value   p-value
treatrIFN-g   -1.184     0.3407  -3.476 0.0005085
[1] "Estimates from the dispersion model"
       Estimate Std. Error
center  0.02986     0.1572
id      1.00235     0.5089
       -2h0  -2*hp -2*p_v(hp) -2*p_b,v(hp)
[1,] 603.30 853.66     692.63       692.95
       cAIC   mAIC   rAIC
[1,] 684.92 698.63 696.95
```

For testing the absence of a random component ($\alpha_1 = 0$ or $\alpha_2 = 0$), we use the deviance based on REMPL, $-2p_{\beta,v}(h_p)$, and fit the following four models including the Cox model and three lognormal frailty models using HL(1,1) method,

M1: Cox model without frailty $(\alpha_1 = 0, \alpha_2 = 0)$ has $-2p_{\beta,v}(h_p) = 707.48$,

M2: model without patient effect $(\alpha_1 > 0, \alpha_2 = 0)$ has $-2p_{\beta,v}(h_p) = 703.66$,

M3: model without center effect $(\alpha_1 = 0, \alpha_2 > 0)$ has $-2p_{\beta,v}(h_p) = 692.99$, and

M4: multilevel model above requiring both patient and center effects $(\alpha_1 > 0, \alpha_2 > 0)$ has $-2p_{\beta,v}(h_p) = 692.95$.

The deviance difference between M3 and M4 is $692.99 - 692.95 = 0.04$, which is not significant at a 5% level ($\chi^2_{1,0.10} = 2.71$), indicating the absence of the random-center effects, i.e. $\alpha_1 = 0$. The deviance difference between M2 and M4 is 703.66-692.95=10.71, indicating that the random-patient effects are necessary, i.e. $\alpha_2 > 0$. In addition, the deviance difference between M1 and M3 is 707.48-692.99=14.49, indicating that the random-patient effects are indeed necessary with or without random-center effects.

Let us choose a final model using information criteria. For M1 we have cAIC=708.68, mAIC=708.68 and rAIC=707.48; for M2 cAIC=702.96, mAIC=706.88 and rAIC=705.66; for M3 cAIC=684.84, mAIC=696.68 and rAIC=694.99; for M4 cAIC=684.92, mAIC=698.63 and rAIC=696.95. All of the three criteria choose M3 in the CGD data set.

Comparison of results with alternative procedures

Using the examples in the previous section, we compare the outcomes from **frailtyHL** and other packages. We consider the three functions (`coxph()`, `coxme()` and `phmm()`) for the lognormal frailty model and the `coxph()` function for the gamma frailty model.

Example 1: Rat data
The codes of `coxph()`, `coxme()` and `phmm()` for fitting lognormal frailty model are, respectively, as follows:

```
>coxph(Surv(time, status) ~ rx +
+         frailty(litter, dist = "gauss"),
+       method = "breslow", rats)

> coxme(Surv(time, status) ~ rx + (1|litter),
+        ties = "breslow", rats)

> phmm(Surv(time, status) ~ rx + (1|litter),
+       data = rats, Gbs = 2000, Gbsvar = 3000,
+       VARSTART = 1, NINIT = 10,
+       MAXSTEP = 200, CONVERG=90)
```

Table 3 summarizes the estimation results. Even though cluster size $n_i \equiv 3$ is not large, the results are similar. For example, MPL and ML estimates for α from `coxme()` and `phmm()` are somewhat different in Table 2 when $n_i \equiv 2$, while they become similar in Table 3.

Next, the code of `coxph()` for fitting the gamma frailty model is below:

```
> coxph(Surv(time, status) ~ rx +
+            frailty(litter, dist = "gamma"),
+        method = "breslow", rats)
```

The results of `frailtyHL()` (HL(0,2), HL(1,2)) and `coxph()` for gamma frailty are also presented in Table 3. For the estimation of β both results from `frailtyHL()` and `coxph()` are similar, but for α they are somewhat different. That is, our REMPL estimates from `frailtyHL()` ($\hat{\alpha} = 0.575$ with HL(0,2) and $\hat{\alpha} = 0.576$ with HL(1,2)) are somewhat larger than the ML estimates from `coxph()` ($\hat{\alpha} = 0.474$ with Breslow method and $\hat{\alpha} = 0.499$ with Efron method).

Table 3: Comparison of different estimation methods for the rat data

Method	Rx $\hat{\beta}$ (SE)	Litter $\hat{\alpha}$ (SE)
lognormal model		
HL(0,1)	0.906 (0.323)	0.427 (0.423)
HL(1,1)	0.911 (0.323)	0.427 (0.423)
`coxph()` (Breslow)	0.905 (0.322)	0.395 (–)
`coxph()` (Efron)	0.913 (0.323)	0.412 (–)
`coxme()` (Breslow)	0.905 (0.322)	0.406 (–)
`coxme()` (Efron)	0.913 (0.323)	0.426 (–)
`phmm()`	0.920 (0.326)	0.449 (–)
gamma model		
HL(0,2)	0.908 (0.324)	0.575 (0.598)
HL(1,2)	0.913 (0.324)	0.576 (0.598)
`coxph()` (Breslow)	0.906 (0.323)	0.474 (–)
`coxph()` (Efron)	0.914 (0.323)	0.499 (–)

Example 2: CGD data

The code of the `coxme()` function for fitting multilevel lognormal frailty model is as follows:

```
> coxme(Surv(tstop - tstart,status) ~
+       treat + (1|center) + (1|id),
+       ties = "breslow", cgd)
```

The results of `frailtyHL()`(HL(0,1), HL(1,1)) and `coxme()` are summarized in Table 4. The results from HL and PPL methods for frailty parameters become more similar because the cluster sizes (the number of patients from different centers) are somewhat large, ranging from 4 to 26.

Table 4: Comparison of different estimation methods for the CGD data

Method	Treat $\hat{\beta}$ (SE)	Center $\hat{\alpha}_1$ (SE)	Patient $\hat{\alpha}_2$ (SE)
lognormal model			
HL(0,1)	-1.074 (0.335)	0.026 (0.153)	0.982 (0.501)
HL(1,1)	-1.184 (0.341)	0.030 (0.157)	1.002 (0.509)
`coxme()` (Breslow)	-1.074 (0.333)	0.033 (–)	0.939 (–)
`coxme()` (Efron)	-1.074 (0.333)	0.032 (–)	0.947 (–)

Summary

The h-likelihood method offers novel possibilities to fit various models with random effects. The **frailtyHL** package for frailty models eliminates the nuisance parameters λ_0 in the h-likelihood (3) by profiling. Such models have important applications in multi-center clinical study (Vaida and Xu, 2000), meta analysis (Rondeau et al., 2008), and genetic analysis (Lee et al., 2006). Therefore, this package can be potentially adopted by statisticians in several different fields.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2009-0088978, No. 2010-0021165 and No. 2010-0011372).

Bibliography

M. Alam, L. Ronnegard, and X. Shen. *hglm: hierarchical generalized linear models*, 2010. URL `http://CRAN.R-project.org/package=hglm`. R package version 1.1.1. [p28]

N. E. Breslow. Discussion of Professor Cox's paper. *Journal of the Royal Statistical Society Series B*, 34: 216–217, 1972. [p29]

N. E. Breslow. Covariance analysis of censored survival data. *Biometrics*, 30:89–99, 1974. [p30]

D. R. Cox and N. Reid. Parameter orthogonality and approximate conditional inference (with discussion). *Journal of the Royal Statistical Society Series B*, 49:1–39, 1987. [p30]

L. Duchateau and P. Janssen. *The Frailty Model*. Springer-Verlag: New York, 2008. [p28]

M. Donohue, R. Overholser, R. Xu, and F. Vaida. Conditional Akaike information under generalized linear and proportional hazards mixed models. *Biometrika*, 98:685–700, 2011. [p33]

M. Donohue and R. Xu. *phmm: proportional hazards mixed-effects model*, 2012. URL `http://CRAN.R-project.org/package=phmm`. R package version 0.7-4. [p28, 30, 31]

T. R. Fleming and D. P. Harrington. *Counting Processes and Survival Analysis*. Wiley: New York, 1991. [p29, 33]

J. R. Gonzalez, V. Rondeau Y. Mazroui, A. Mauguen, and A. Diakité. *frailtypack: frailty models using a semi-parametrical penalized likelihood estimation or a*

parametrical estimation, 2012. URL http://CRAN.R-project.org/package=frailtypack. R package version 2.2-23. [p28]

I. D. Ha and Y. Lee. Estimating frailty models via Poisson hierarchical generalized linear models. *Journal of Computational and Graphical Statistics*, 12: 663–681, 2003. [p28, 30]

I. D. Ha and Y. Lee. Comparison of hierarchical likelihood versus orthodox best linear unbiased predictor approaches for frailty models. *Biometrika*, 92: 717–723, 2005. [p28, 30]

I. D. Ha, Y. Lee and G. MacKenzie. Model selection for multi-component frailty models. *Statistics in Medicine*, 26:4790–4807, 2007. [p29, 33]

I. D. Ha, Y. Lee and J. K. Song. Hierarchical likelihood approach for frailty models. *Biometrika*, 88: 233–243, 2001. [p28, 29, 30]

I. D. Ha, M. Noh and Y. Lee. Bias reduction of likelihood estimators in semi-parametric frailty models. *Scandinavian Journal of Statistics*, 37:307–320, 2010. [p30, 31, 32]

I. D. Ha, M. Noh and Y. Lee. *frailtyHL: frailty models using h-likelihood*, 2012. URL http://CRAN.R-project.org/package=frailtyHL. R package version 1.1. [p28, 30]

I. D. Ha, T. Park and Y. Lee. Joint modelling of repeated measures and survival time data. *Biometrical Journal*, 45:647–658, 2003. [p32]

I. D. Ha, R. Sylvester, C. Legrand and G. MacKenzie. Frailty modelling for survival data from multicentre clinical trials. *Statistics in Medicine*, 30:2144–2159, 2011. [p31, 32]

P. Hougaard. *Analysis of Multivariate Survival Data*. Springer-Verlag: New York, 2000. [p28, 29]

Y. Lee and I. D. Ha. Orthodox BLUP versus h-likelihood methods for inferences about random effects in Tweedie mixed models. *Statistics and Computing*, 20:295–303, 2010. [p31]

Y. Lee, M. Jang and W. Lee. Prediction interval for disease mapping using hierarchical likelihood. *Computational Statistics*, 26:159–179, 2011. [p31]

Y. Lee and J. A. Nelder. Hierarchical generalized linear models (with discussion). *Journal of the Royal Statistical Society Series B*, 58:619–678, 1996. [p28, 30]

Y. Lee and J. A. Nelder. Hierarchical generalised linear models: a synthesis of generalised linear models, random-effect models and structured dispersions. *Biometrika*, 88:987–1006, 2001. [p30]

Y. Lee and J. A. Nelder. Double hierarchical generalised linear models (with discussion). *Applied Statistics*, 55:139–185, 2006. [p32]

Y. Lee, J. A. Nelder and Y. Pawitan. *Generalised Linear Models with Random Effects: Unified Analysis via H-likelihood*. Chapman & Hall: London, 2006. [p28, 30, 31, 32, 33, 35]

N. Mantel, N. R. Bohidar and J. L. Ciminera. Mantel-Haenszel analyses of litter-matched time-to-response data, with modifications for recovery of interlitter information. *Cancer Research*, 37:3863–3868, 1977. [p32]

C. A. McGilchrist and C. W. Aisbett. Regression with frailty in survival analysis. *Biometrics*, 47:461–466, 1991. [p31]

M. Molas. *HGLMMM: hierarchical generalized linear models*, 2010. URL http://CRAN.R-project.org/package=HGLMMM. R package version 0.1.1. [p28]

M. Noh and Y. Lee. *dhglm: double hierarchical generalized linear models*, 2011. URL http://CRAN.R-project.org/package=dhglm. R package version 1.0. [p28]

S. Ripatti and J. Palmgren. Estimation of multivariate frailty models using penalized partial likelihood. *Biometrics*, 56:1016–1022, 2000. [p30]

V. Rondeau, S. Michiels, B. Liquet and J. P. Pignon. Investigating trial and treatment heterogeneity in an individual patient data meta-analysis of survival data by means of the penalized maximum likelihood approach. *Statistics in Medicine*, 27:1894-910, 2008. [p35]

Y. Sakamoto, M. Ishiguro and G. Kitagawa. *Akaike Information Criterion Statistics*. KTK Scientific Publisher: Tokyo, 1986. [p33]

T. M. Therneau. *survival: survival analysis, including penalised likelihood*, 2010. URL http://CRAN.R-project.org/package=survival. R package version 2.36-2. [p28, 30, 31]

T. M. Therneau. *coxme: Mixed Effects Cox Models*, 2011. URL http://CRAN.R-project.org/package=coxme. R package version 2.2-1. [p28, 30]

T. M. Therneau and P. M. Grambsch. *Modelling Survival Data: Extending the Cox Model*. Springer-Verlag: New York, 2000. [p29, 30, 31]

F. Vaida and R. Xu. Proportional hazards model with random effects. *Statistics in Medicine*, 19:3309–3324, 2000. [p32, 35]

K. K. W. Yau. Multilevel models for survival analysis with random effects. *Biometrics*, 57:96–102, 2001. [p29]

Il Do Ha
Department of Asset Management
Daegu Haany University, Korea
idha@dhu.ac.kr

Maengseok Noh
Department of Statistics
Pukyong National University, Korea
msnoh@pknu.ac.kr

Youngjo Lee
Department of Statistics
Seoul National University, Korea
youngjo@snu.ac.kr

influence.ME: Tools for Detecting Influential Data in Mixed Effects Models

by Rense Nieuwenhuis, Manfred te Grotenhuis, and Ben Pelzer

Abstract **influence.ME** provides tools for detecting influential data in mixed effects models. The application of these models has become common practice, but the development of diagnostic tools has lagged behind. **influence.ME** calculates standardized measures of influential data for the point estimates of generalized mixed effects models, such as DFBETAS, Cook's distance, as well as percentile change and a test for changing levels of significance. **influence.ME** calculates these measures of influence while accounting for the nesting structure of the data. The package and measures of influential data are introduced, a practical example is given, and strategies for dealing with influential data are suggested.

The application of mixed effects regression models has become common practice in the field of social sciences. As used in the social sciences, mixed effects regression models take into account that observations on individual respondents are nested within higher-level groups such as schools, classrooms, states, and countries (Snijders and Bosker, 1999), and are often referred to as multilevel regression models. Despite these models' increasing popularity, diagnostic tools to evaluate fitted models lag behind.

We introduce **influence.ME** (Nieuwenhuis, Pelzer, and te Grotenhuis, 2012), an R-package that provides tools for detecting influential cases in mixed effects regression models estimated with **lme4** (Bates and Maechler, 2010). It is commonly accepted that tests for influential data should be performed on regression models, especially when estimates are based on a relatively small number of cases. However, most existing procedures do not account for the nesting structure of the data. As a result, these existing procedures fail to detect that higher-level cases may be influential on estimates of variables measured at specifically that level.

In this paper, we outline the basic rationale on detecting influential data, describe standardized measures of influence, provide a practical example of the analysis of students in 23 schools, and discuss strategies for dealing with influential cases. Testing for influential cases in mixed effects regression models is important, because influential data negatively influence the statistical fit and generalizability of the model. In social science applications of mixed models the testing for influential data is especially important, since these models are frequently based on large numbers of observations at the individual level while the number of higher level groups is relatively small. For instance, Van der Meer, te Grotenhuis, and Pelzer (2010) were unable to find any country-level comparative studies involving more than 54 countries. With such a relatively low number of countries, a single country can easily be overly influential on the parameter estimates of one or more of the country-level variables.

Detecting Influential Data

All cases used to estimate a regression model exert some level of influence on the regression parameters. However, if a single case has extremely high or low scores on the dependent variable relative to its expected value — given other variables in the model, one or more of the independent variables, or both — this case may overly influence the regression parameters by 'pulling' the estimated regression line towards itself. The simple inclusion or exclusion of such a single case may then lead to substantially different regression estimates. This runs against distributional assumptions associated with regression models, and as a result limits the validity and generalizability of regression models in which influential cases are present.

The analysis of residuals cannot be used for the detection of influential cases (Crawley, 2007). Cases with high *residuals* (defined as the difference between the observed and the predicted scores on the dependent variable) or with high standardized residuals (defined as the residual divided by the standard deviation of the residuals) are indicated as outliers. However, an influential case is not always an outlier. On the contrary: a strongly influential case dominates the regression model in such a way, that the estimated regression line lies closely to this case. Although influential cases thus have extreme values on one or more of the variables, they can be *onliers* rather than *outliers*. To account for this, the *(standardized) deleted residual* is defined as the difference between the observed score of a case on the dependent variable, and the predicted score from the regression model that is based on data from which that case was removed.

Just as influential cases are not necessarily outliers, outliers are not necessarily influential cases. This also holds for deleted residuals. The reason for this is that the amount of influence a case exerts on the regression slope is not only determined by how well its (observed) score is fitted by the specified regression model, but also by its score(s) on the

independent variable(s). The degree to which the scores of a case on the independent variable(s) are extreme is indicated by the *leverage* of this case. A higher leverage means more extreme scores on the independent variable(s), and a greater potential of overly influencing the regression outcomes. However, if a case has very extreme scores on the independent variable(s) but is fitted very well by a regression model, and if this case has a low deleted (standardized) residual, this case is not necessarily overly influencing the outcomes of the regression model.

Since neither outliers, nor cases with a high leverage, are necessarily influential, a different procedure is required for detecting influential cases. The basic rationale behind measuring influential cases is based on the principle that when single cases are iteratively omitted from the data, models based on these data should not produce substantially different estimates. If the model parameters change substantially after a single case is excluded, this case may be regarded as too influential. However, how much change in the model parameters is acceptable? To standardize the assessment of how influential a single case is, several measures of influence are commonly used. First, DFBETAS is a standardized measure of the absolute difference between the estimate with a particular case included and the estimate without that particular case (Belsley, Kuh, and Welsch, 1980). Second, Cook's distance provides an overall measurement of the change in all parameter estimates, or a selection thereof (Cook, 1977). In addition, we introduce the measure of percentile change and a test for changing levels of significance of the fixed parameters.

Up to this point, this discussion on influential data was limited to how single cases can overly influence the point estimates (or BETAS) of a regression model. Single cases, however, can also bias the confidence intervals of these estimates. As indicated above, cases with high leverage can be influential because of their extreme values on the independent variables, but not necessarily are. Cases with high leverage but a low deleted residual compress standard errors, while cases with low leverage and a high deleted residual inflate standard errors. Inferences made to the population from models in which such cases are present may be incorrect.

Detecting Influential Data in Mixed Effects Models

Other options are available in R that help evaluating the fit of regression models, including the detection of influential data. The base R installation provides various plots for regression models, including but not limited to plots showing residuals versus the fitted scores, Cook's distances, and the leverage versus the deleted residuals. The latter plot can be used to detect cases that affect the inferential properties of the model, as discussed above. These plots, however, are not available for mixed effects models. The **LMERConvenienceFunctions** package provides model criticism plots, including the density of the model residuals and the fitted values versus the standardized residuals (Tremblay, 2012). However, while this package works with the **lme4** package, it only is applicable for *linear* mixed effects models.

The **influence.ME** package introduced here contributes to these existing options, by providing several measures of influential data for *generalized* mixed effects models. The limitation is that, unfortunately, as far as we are aware, the measure of leverage was not developed for generalized mixed effects models. Consequently, the current installment of **influence.ME** emphasizes detecting the influence of cases on the point estimates of generalized mixed effect models. It does, however, provide a basic test for detecting whether single cases change the level of significance of an estimate, and therefore the ability to make inferences from the estimated model.

To apply the logic of detecting influential data to generalized mixed effects models, one has to measure the influence of a particular higher level group on the estimates of a predictor measured at that level. The straightforward way is to delete all observations from the data that are nested within a single higher level group, then re-estimate the regression model, and finally evaluate the change in the estimated regression parameters. This procedure is then repeated for each higher-level group separately.

The `influence` function in the **influence.ME** package performs this procedure automatically, and returns an object containing information on the parameter estimates excluding the influence of each higher level group separately. The returned object of class `"estex"` (`EST`imates `EX`cluding the influence of a group) can then be passed on to one of the functions calculating standardized measures of influence (such as DFBETAS and Cook's Distance, discussed in more detail in the next section). Since the procedure of the `influence` function entails re-estimating mixed effects models several times, this can be computationally intensive. Unlike the standard approach in R, we separated the estimation procedure from calculating the measures of influence themselves. This allows the user to process a single model once using the `influence` function, and then to evaluate it using various measures and plots of influence.

In detecting influential data in mixed effects models, the key focus is on changes in the estimates of variables measured at the group-level. However, most mixed effects regression models estimate the effects of both lower-level and higher-level variables simultaneously. Langford and Lewis (1998) developed a procedure in which the mixed effects model is modified to neutralize the group's influence on the higher-level estimate, while at the same time allowing the lower-level observations nested within

that group to help estimate the effects of the lower-level predictors in the model. For each higher-level unit evaluated based on this method, the intercept-vector of the model is set to 0, and an (additional) dummy variable is added to the model, with score 1 for the respective higher level case. This way, the case under investigation does not contribute to the variance of the random intercept, nor to the effects of variables measured at the group-level. **influence.ME** provides this functionality, which is accessed by specifying `delete=FALSE` as an option to the `influence` function. As a result of the specific modification of the model-specification, this specific procedure suggested by Langford and Lewis (1998) does not work when factor-variables are used in the regression model.

Finally, **influence.ME** also allows for detecting the influence of lower-level cases in the mixed effects model. In social science applications of mixed effects models, with a great number of lower-level observations nested in a limited number of groups, this will not always be feasible. Detecting influence of lower-level observations is supported for applications in various disciplines where mixed effects models are typically applied to only a limited number of observations per group. This procedure is accessed by specifying `obs=TRUE` as an option to the `influence` function. The `influence` function can either determine the influence of higher-level cases, or of single observations, but not both at the same time.

The Outcome Measures

The `influence` function described above returns an object with information on how much the parameter estimates in a mixed effects model change, after the (influence of) observations of higher-level groups and their individual-level observations were removed from it iteratively. This returned object can then be passed on to functions that calculate standardized measures of influence. **influence.ME** offers four such measures, which are detailed in this section.

DFBETAS

DFBETAS is a standardized measure that indicates the level of influence observations have on single parameter estimates (Fox, 2002). Regarding mixed models, this relates to the influence a higher-level unit has on the parameter estimate. DFBETAS is calculated as the difference in the magnitude of the parameter estimate between the model including and the model excluding the higher level case. This absolute difference is divided by the standard error of the parameter estimate excluding the higher level unit under investigation:

$$DFBETAS_{ij} = \frac{\hat{\gamma}_i - \hat{\gamma}_{i(-j)}}{se\left(\hat{\gamma}_{i(-j)}\right)}$$

in which i refers to the parameter estimate, and j the higher-level group, so that $\hat{\gamma}_i$ represents the original estimate of parameter i, and $\hat{\gamma}_{i(-j)}$ represents the estimate of parameter i, after the higher-level group j has been excluded from the data.

In **influence.ME**, values for DFBETAS in mixed effects models can be calculated using the function `dfbetas`, which takes the object returned from `influence` as input. Further options include `parameters` to provide a vector of index numbers or names of the selection of parameters for which DFBETAS is to be calculated. The default option of `dfbetas` is to calculate DFBETAS for estimates of all fixed effects in the model.

As a rule of thumb, a cut-off value is given for DFBETAS (Belsley et al., 1980):

$$2/\sqrt{n}$$

in which n, the number of observations, refers to the number of groups in the grouping factor under evaluation (and not to the number of observations nested within the group under investigation). Values exceeding this cut-off value are regarded as overly influencing the regression outcomes for that specific estimate.

Cook's Distance

Since DFBETAS provides a value for each parameter and for each higher-level unit that is evaluated, this often results in quite a large number of values to evaluate (Fox, 2002). An alternative is provided by Cook's distance, a commonly used measure of influence. Cook's distance provides a summary measure for the influence a higher level unit exerts on *all* parameter estimates simultaneously, or a selection thereof. A formula for Cook's Distance is provided (Snijders and Bosker, 1999; Snijders and Berkhof, 2008):

$$C_j^{0F} = \frac{1}{r+1}\left(\hat{\gamma} - \hat{\gamma}_{(-j)}\right)' \widehat{\Sigma}_F^{-1}\left(\hat{\gamma} - \hat{\gamma}_{(-j)}\right)$$

in which $\hat{\gamma}$ represents the vector of original parameter estimates, $\hat{\gamma}_{(-j)}$the parameter estimates of the model excluding higher-level unit j, and $\widehat{\Sigma}_F$ represents the covariance matrix. In **influence.ME**, the covariance matrix of the model *excluding* the higher-level unit under investigation j is used. Finally, r is the number of parameters that are evaluated, excluding the intercept vector.

As a rule of thumb, cases are regarded as too influential if the associated value for Cook's Distance exceeds the cut-off value of (Van der Meer et al., 2010):

$$4/n$$

in which n refers to the number of groups in the grouping factor under evaluation.

In **influence.ME**, values for Cook's distance in mixed effects models are calculated using the function `cooks.distance`, which takes the object returned from `influence` as input. Further options include `parameters` to provide a vector of index numbers or names of the parameters for which Cook's Distance is to be calculated. In addition, the user can specify `sort=TRUE` to have the values for Cook's distance returned in descending order.

As a final note, it is pointed out that if Cook's distance is calculated based on a single parameter, the Cook's distance equals the squared value of DFBETAS for that parameter. This is also reflected in their respective cut-off values:

$$\sqrt{\frac{4}{n}} = \frac{2}{\sqrt{n}}$$

Percentile Change

Depending upon the goal for which the mixed model is estimated (prediction vs. hypothesis testing), the use of formal measures of influence as DFBETAS and Cook's distance may be less desirable. The reason for this is that based on these measures it is not immediately clear to what extent parameter estimates change. For substantive interpretation of the model outcomes, the relative degree to which a parameter estimate changes may provide more meaningful information. A simple alternative is therefore offered by the function `pchange`, which takes the same input-options as the `dfbetas` function. For each higher-level group, the percentage of change is calculated as the absolute difference between the parameter estimate both including and excluding the higher-level unit, divided by the parameter estimate of the complete model and multiplied by 100%. A percentage of change is returned for each parameter separately, for each of the higher-level units under investigation. In the form of a formula:

$$\left(\hat{\gamma} - \hat{\gamma}_{(-j)}\right) \frac{1}{\hat{\gamma}} * 100\%$$

No cut-off value is provided, for determining what percent change in parameter estimate is considered too large will primarily depend on the goal for which the model was estimated and, more specifically, the nature of the hypotheses that are tested.

Test for changes in significance

As discussed above, even when cases are not influential on the point estimates (BETAS) of the regression model, cases can still influence the standard errors of these estimates. Although **influence.ME** cannot provide the leverage measure to detect this, it provides a test for changes in the statistical significance of the fixed parameters in the mixed effects model.

The `sigtest` function tests whether excluding the influence of a single case changes the statistical significance of any of the variables in the model. This test of significance is based on the test statistic provided by the **lme4** package. The nature of this statistic varies between different distributional families in the generalized mixed effects models. For instance, the t-statistic is related to a normal distribution while the z-statistic is related to binomial distributions.

For each of the cases that are evaluated, the test statistic of each variable is compared to a test-value specified by the user. For the purpose of this test, the parameter is regarded as statistically significant if the test statistic of the model exceeds the specified value. The `sigtest` function reports for each variable the estimated test statistic after deletion of each evaluated case, whether or not this updated test statistic results in statistical significance based on the user-specified value, and whether or not this new statistical significance differs from the significance in the original model. So, in other words, if a parameter was statistically significant in the original model, but is no longer significant after the deletion of a specific case from the model, this is indicated by the output of the `sigtest` function. It is also indicated when an estimate was not significant originally, but reached statistical significance after deletion of a specific case.

Plots

All four measures of influence discussed above, can also be plotted. The `plot` function takes the output of the `influence` function to create a dotplot of a selected measure of influence (cf. Sarkar, 2008). The user can specify which measure of influence is to be plotted using the `which=` option. The `which=` option defaults to `"dfbetas"`. Other options are to select `"cook"` to plot the Cook's distances, `"pchange"` to plot the percentage change, and `"sigtest"` to plot the test statistic of a parameter estimate after deletion of specific cases.

All plots allow the output to be sorted (by specifying `sort=TRUE` and the variable to sort on using `to.sort=` (not required for plotting Cook's distances). In addition, a cut-off value can be specified using (`cutoff=`). Values that exceed this cut-off value will be plotted visually differently, to facilitate the identification of influential cases. By default, the results for all cases and all variables are plotted, but a selection of these can be made by specifying `parameters=` and / or `groups=`. Finally, by specifying `abs=TRUE` the absolute values of the measure of influence are plotted.

Example: students in 23 schools

In our example, we are interested in the relationship between the degree of structure that schools attempt to enforce in their classrooms and students' performance on a math test. Could it be that a highly structured class affects their performance?

The **influence.ME** package contains the `school23` data.frame, that provides information on the performance of 519 students in 23 schools. Measurements include individual students' score on a math test, school-level measurements of class structure, and several additional independent variables. Student's class and school are equivalent in this data, since only one class per school is available. These data are a subset of the NELS-88 data (National Education Longitudinal Study of 1988). The data are publicly available from the internet: `http://www.ats.ucla.edu/stat/examples/imm/`, and are reproduced with kind permission of Ita Kreft and Jan de Leeuw (1998).

First, using the **lme4** package, we estimate a multivariate mixed effects model with students nested in schools, a random intercept, a measurement of individual students' time spent on math homework, and a measurement of class structure at the school level. For the purpose of our example, we assume here that the math, homework, and structure variables were correctly measured at the interval level.

```
library(influence.ME)
data(school23)

school23 <- within(school23,
   homework <- unclass(homework))

m23 <- lmer(math ~ homework + structure
   + (1 | school.ID),
   data=school23)

print(m23, cor=FALSE)
```

This results in the summary of the model based on 23 schools (assigned to object `m23`), as shown below.

```
Linear mixed model fit by REML
Formula: math ~ homework +
   structure + (1 | school.ID)
   Data: school23
  AIC  BIC logLik deviance REMLdev
 3734 3756  -1862     3728    3724
Random effects:
 Groups    Name        Variance Std.Dev.
 school.ID (Intercept) 19.543   4.4208
 Residual              71.311   8.4446
Number of obs: 519, groups: school.ID, 23

Fixed effects:
            Estimate Std. Error t value
(Intercept)  52.2356     5.3940   9.684
homework      2.3938     0.2771   8.640
structure    -2.0950     1.3237  -1.583
```

Based on these estimates, we may conclude that students spending more time on their math homework score better on a math test. Regarding class structure, however, we do not find a statistically significant association with math test scores. But, can we now validly conclude that class structure does not influence students' math performance, based on the outcomes of this model?

Visual Examination

Since the analysis in the previous section has been based on the limited number of 23 schools, it is, of course, possible that observations on single schools have overly influenced these findings. Before using the tools provided in the **influence.ME** package to formally evaluate this, a visual examination of the relationship between class structure and math test performance, aggregated to the school level, will be performed.

```
struct <- unique(subset(school23,
    select=c(school.ID, structure)))

struct$mathAvg <- with(school23,
    tapply(math, school.ID, mean))

dotplot(mathAvg ~ factor(structure),
    struct,
    type=c("p","a"),
    xlab="Class structure level",
    ylab="Average Math Test Score")
```

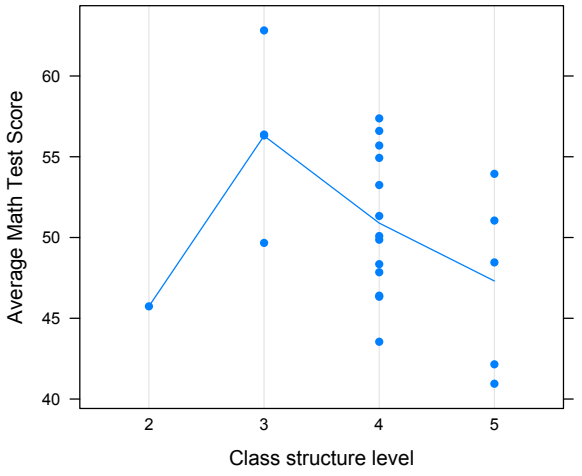


Figure 1: Visual examination of class structure and math performance

In the syntax above, a bivariate plot of the aggregated math scores and class structure is created, which is shown in Figure 1. In this plot, it is clear that one single school represented in the lower-left corner of the graph seems to be an outlier, and - more importantly - the non-linear curve shown in this graph clearly indicates this single school with class structure level of 2 may overly influence a linear regression line estimated based on these data.

Calculating measures of influence

In the previous section, based on Figure 1 we suspected that the combination in one specific school of the low average math test results of students, and the low level of class structure in that school, may have overly influenced the original analysis of our data. However, this is only a bivariate examination of the data, and therefore does not take into account other variables in the model. Hence, in our example, our preliminary conclusion that this may be an influential case is not controlled for possible effects of the homework variable. A better test is provided by standardized measures of influence, as calculated from the regression model rather than from the raw data.

The first step in detecting influential data is to determine the extent to which the parameter estimates in model `m23` change, when iteratively each of the schools is deleted from the data. This is done with the `influence` function:

```
estex.m23  <- influence(m23, "school.ID")
```

The `influence` function takes a mixed effects regression model as input (here: `m23`), and the grouping factor needs to be specified, which in our case is `school.ID`. We assign the output of the `influence` function to an object named `estex.m23`. Below, we use this object as input to the `dfbetas` function, to calculate DFBETAS.

```
dfbetas(estex.m23,
    parameters=c(2,3))
```

This results in a substantial amount of output, a portion of which is shown below. Only the DFBETAS for the `homework` and `structure` variables were returned, since `parameters=c(2,3)` was specified.

```
         homework    structure
6053  -0.13353732 -0.168139487
6327  -0.44770666  0.020481057
6467   0.21090081  0.015320965
7194  -0.44641247  0.036756281
7472  -0.55836772  1.254990963
...
72292  0.62278508  0.003905031
72991  0.52021424  0.021630219
```

The numerical output given above by the `dfbetas` function provides a detailed report of the values of DFBETAS in the model. For each variable, as well as for each nesting group (in this example: each school), a value for DFBETAS is computed and reported upon. The cut-off value of DFBETAS equals $2/\sqrt{n}$ (Belsley et al., 1980), which in this case equals $2/\sqrt{23} = .41$. The estimate for class structure in this model seems to be influenced most strongly by observations in school number 7472. The DFBETAS for the `structure` variable clearly exceeds the cut-off value of .41. Also, the estimates of the `homework` variable changes substantially with the deletion of several schools, as indicated by the high values of DFBETAS.

A plot (shown in Figure 2) of the DFBETAS is created using:

```
plot(estex.m23,
    which="dfbetas",
    parameters=c(2,3),
    xlab="DFbetaS",
    ylab="School ID")
```

Based on Figure 2, it is clear that both the `structure` and the `homework` variables are highly susceptible to the influence of single schools. For the `structure` variable this is not all that surprising, since class structure was measured at the school level and shown in Figure 1 to be very likely to be influenced by a single case: school number 7472. The observation that high values of DFBETAS were found for the `homework` variable, suggests that substantial differences between these schools exist in terms of how much time students spend on average on their homework. Therefore, we suggest that in mixed effects regression models, both the estimates of individual-level and group-level variables are evaluated for influential data.

The measure of Cook's distance allows to determine the influence a single higher-level group has on the estimates of multiple variables simultaneously. So, since the `cooks.distance` function allows to specify a selection of variables on which the values for Cook's distance are to be calculated, this can be used to limit the evaluation to the measurements at the group-level exclusively. Note, that whereas DFBETAS always relates to single variables, Cook's distance is a summary measure of changes on all parameter estimates it is based on. Reports on Cook's distance thus should always specify on which variables these values are based.

To continue our example, we illustrate the `cooks.distance` function on a single variable, since class structure is the only variable measured at the school-level. In the example below, we use the same object that was returned from the `influence` function. The specification of this function is similar to `dfbetas`, and to create a plot of the Cook's distances we again use the `plot` function with the spec-

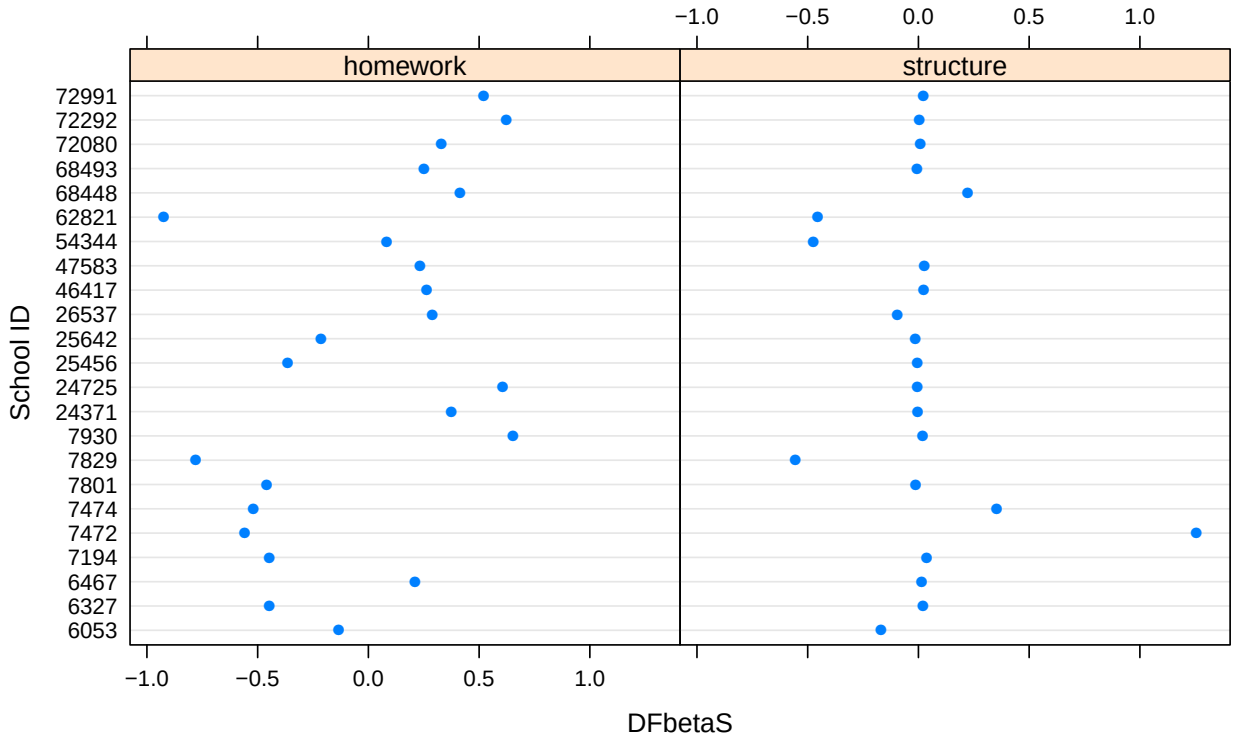


Figure 2: DFBETAS of class structure and homework

ification `which="cook"`. We specify two additional arguments to augment the figure. First, we specify `sort=TRUE` to have the resulting Cook's distances sorted in a descending order in the figure. The appropriate cut-off value for Cook's distance with 23 nesting groups equals to $4/23 = .17$. By specifying the cut-off value with `cutoff=.17`, Cook's distances exceeding the specified value are easily identified in the resulting figure. Thus, to receive both numeric output and a graphical representation (Figure 3), the following specification of `cooks.distance` and `plot` is given:

```
cooks.distance(estex.m23,
    parameter=3, sort=TRUE)

plot(estex.m23, which="cook",
    cutoff=.17, sort=TRUE,
    xlab="Cook´s Distance",
    ylab="School ID")
```

The output below shows one value of Cook's distance for each nesting group, in this case for each school.

```
              [,1]
24371 6.825871e-06
72292 1.524927e-05
...
54344 2.256612e-01
7829  3.081222e-01
7472  1.575002e+00
```

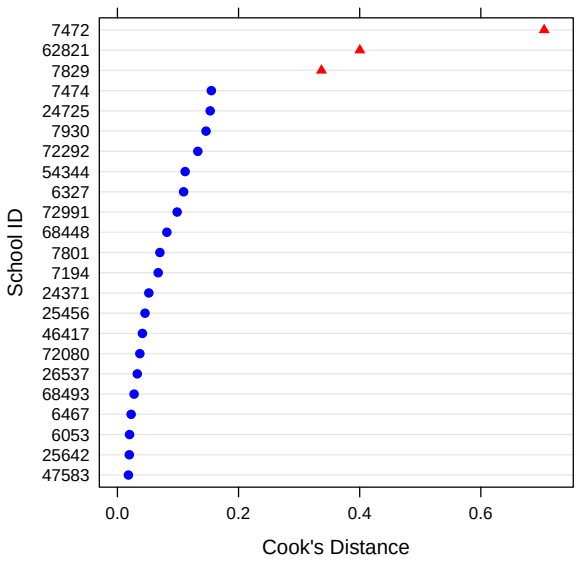


Figure 3: Cook's Distance based on class structure

Only a selection of the output is shown here. A

few schools exceed the cut-off value (in Figure 3 these are indicated with red triangles), but one school stands out: 7472. Clearly, this school strongly influences the outcomes regarding the `structure` variable, as we already suspected based on our bivariate visual examination in Figure 1.

Testing for Changes in Statistical Significance (sigtest)

In the example below, the `sigtest` function is used to test for changing levels of significance after deletion of each of the 23 schools from our example model. We are specifically interested in the level of significance of the `structure` variable, for which it was already established above that school with number 7472 is very influential. Since we observed a negative effect in the original model, we specify `test=-1.96` to test for significance at a commonly used value (-1.96) of the test statistic. Note that since we estimated a normally distributed model, the test statistic here is the t-value.

```
sigtest(estex.m23, test=-1.96)$structure[1:10,]
```

In the example above, we only request the results for the `structure` variable and for the first 10 schools. In the results presented below, three columns are shown. The first column (`Altered.Teststat`) shows the value of the test statistic (here for the `structure` variable) after the deletion of the respective schools (indicated in the row labels). Especially school number 7472 stands out. In the original model, the test statistic for the structure variable was -1.583, which was not significant. When the influence of school number 7472 is excluded from the model, the test statistic now is -2.72, which exceeds the selected value of -1.96 selected by us. That the structure variable would be significant by deletion of school 7472 is indicated in the second column (`Altered.Sig`). The `Changed.Sig` column finally confirms whether the level of significance of the `structure` variable (which was not significant in the original model) changed to significant after each of the schools was deleted.

In the case of our example, the results for Cook's Distance and the results of this test for changing levels of significance both indicate that school number 7472 overly influences the regression outcomes regarding the school-level `structure` variable. Referring to the discussion on influential data above, however, we emphasize that this is not necessarily always the case. Cases can influence the point estimates without affecting their level of significance, or affect the level of significance without overly affecting the point estimate itself. Therefore, both tests should be performed.

```
      Altered.Teststat Altered.Sig Changed.Sig
6053        -1.326409       FALSE       FALSE
6327        -1.688663       FALSE       FALSE
6467        -1.589960       FALSE       FALSE
7194        -1.512686       FALSE       FALSE
7472        -2.715805        TRUE        TRUE
7474        -1.895138       FALSE       FALSE
7801        -1.534023       FALSE       FALSE
7829        -1.045866       FALSE       FALSE
7930        -1.566117       FALSE       FALSE
24371       -1.546838       FALSE       FALSE
```

Before, using DFBETAS, we identified several schools that overly influence the estimate of the `homework` variable. We have there performed `sigtest` test to evaluate whether deletion of any of the schools changes the level of significance of the `homework` variable. These results are not shown here, but indicated that the deletion of none of the schools changed the level of significance of the `homework` variable.

Measuring the influence of lower-level observations

Finally, it is possible that a single lower-level observation affects the results of the mixed effects model, especially for data with a limited number of lower-level observations per group. In our example, this would refer to a single student affecting the estimates of either the individual-level variables, the school-level variables, or both. Here, we test whether one or more individual students affect the estimate of the school-level `structure` variable.

To perform this test, the `influence` function is used, and `obs=TRUE` is specified to indicate that single observations (rather than groups) should be evaluated. The user is warned that this procedure often will be computationally intensive when the number of lower-level observations is large.

Next, we request Cook's Distances specifically for the `structure` variable. Since the number of student-level observations in this model is 519, and cut-off value for Cook's Distance is defined as $4/n$, the cut-off value is $4/519 = .0077$. The resulting output is extensive, since a Cook's Distance is calculated for any of the 519 students. Therefore, in the example below, we directly test which of the resulting Cook's Distances exceeds the cut-off value.

```
estex.obs  <- influence(m23, obs=TRUE)
cks.d <- cooks.distance(estex.obs, parameter=3)
which(cks.d > 4/519)
```

The output is not shown here, but the reader can verify that students with numbers 88 and 89 exert too much influence on the estimate of the structure variable. Using the `sigtest` function, however, showed that the deletion of none of the students from the

data affected the level of significance of the structure variable, nor of any of the other variables in the model.

Dealing with Influential Data

Now that overly influential cases have been identified in our model, we have to decide how to deal with them. Generally, there are several strategies, including getting more data, checking data consistency, adapting model specification, deleting the influential cases from the model, and obtaining additional measurements on existing cases to account for the overly influential cases (Van der Meer et al., 2010; Harrell, Jr., 2001).

Since overly influential data are a problem especially encountered in models based on a limited number of cases, a straightforward remedy would be to observe more cases in the population of interest. In our example, if we would be able to sample more schools, it may very well turn out that we observe several additional schools with a low score on the `structure` variable, so that school number 7472 is no longer influential. Secondly, there may have been measurement, coding, or transcription errors in the data, that have lead to extreme scores on one or more of the variables (i.e. it may be worthwhile, if possible, to check whether class structure and / or students' math performance in school 7472 really is that low). Thirdly, the model specification may be improved. If the data are used to estimate too complex models, or if parameterization is incorrect, influential cases are more likely to occur. Perhaps the `structure` variable should have been treated as categorical.

These are all general strategies, but cannot always be applied. Depending on the research setting, it is not always feasible to obtain more observations, to return to the raw data to check consistency, or to reduce model complexity or change parameterization.

The fourth strategy, deleting influential cases from the model, can often be applied. In general, we suggest deleting influential cases one at the time and then to re-evaluating the model. Deleting one or more influential cases from a mixed effects model is done with the `exclude.influence` function. The input of this function is a mixed effects model object, and it returns an updated mixed effects model from which a specified group was deleted. To illustrate, we delete school number 7472 (which was identified as being overly influential) and its individual-level observations, using the example code below:

```
m22 <- exclude.influence(m23,
    "school.ID", "7472")

print(m22, cor=FALSE)
```

The `exclude.influence` function takes a mixed effects model as input, and requires the specification of the grouping factor (`school.ID`) and the group to be deleted (`7472`). It returns a re-estimated mixed effects model, that we assign to the object `m22`. The summary of that model is shown below:

```
Linear mixed model fit by REML
Formula: math ~ homework + structure
   + (1 | school.ID)
   Data: ..1
  AIC  BIC logLik deviance REMLdev
 3560 3581  -1775     3554    3550
Random effects:
 Groups    Name        Variance Std.Dev.
 school.ID (Intercept) 15.333   3.9157
 Residual              70.672   8.4067
Number of obs: 496, groups: school.ID, 22

Fixed effects:
            Estimate Std. Error t value
(Intercept)  59.4146     5.9547   9.978
homework      2.5499     0.2796   9.121
structure    -3.8949     1.4342  -2.716
```

Two things stand out when this model summary is compared to our original analysis. First, the number of observations is lower (496 versus 519), as well as the number of groups (22 versus 23). More importantly, though, the negative effect of the `structure` variable now is statistically significant, whereas it was not in the original model. So, now these model outcomes indicate that higher levels of class structure indeed are associated with lower math test scores, even when controlled for the students' homework efforts.

Further analyses should repeat the analysis for influential data, for other schools may turn out to be overly influential as well. These repetitive steps are not presented here, but as it turned out, three other schools were overly influential. However, the substantive conclusions drawn based on model `m22` did not change after their deletion.

Finally, we suggest an approach for dealing with influential data, based on Lieberman (2005). He argues that the presence of outliers may indicate that one or more important variables were omitted from the model. Adding additional variables to the model may then account for the outliers, and improve the model fit. We discussed above that an influential case is not necessarily an outlier in a regression model. Nevertheless, if additional variables in the model can account for the fact that an observation has extreme scores on one or more variables, the case may no longer be an influential one.

Thus, adding important variables to the model may solve the problem of influential data. When the

observations in a regression model are, for instance, randomly sampled respondents in a large-scale survey, it often is impossible to return to these respondents for additional measurements. However, in social science applications of mixed effects models, the higher-level groups are often readily accessible cases such as schools and countries. It may very well be possible to obtain additional measurements on these schools or countries, and use these to remedy the presence of influential data.

Summary

influence.ME provides tools for detecting influential data in mixed effects models. The application of these models has become common practice, but the development of diagnostic tools lag behind. **influence.ME** calculates standardized measures of influential data such as DFBETAS and Cook's distance, as well as percentile change and a test for changing in statistical significance of fixed parameter estimates. The package and measures of influential data were introduced, a practical example was given, and strategies for dealing with influential data were suggested.

Bibliography

D. Bates and M. Maechler. *lme4: Linear mixed-effects models using S4 classes*, 2010. URL http://CRAN.R-project.org/package=lme4. R package version 0.999375-35. [p38]

D. Belsley, E. Kuh, and R. Welsch. *Regression Diagnostics. Identifying Influential Data and Sources of Collinearity*. Wiley, 1980. [p39, 40, 43]

R. Cook. Detection of influential observations in linear regression. *Technometrics*, 19(1):15–18, 1977. [p39]

M. J. Crawley. *The R Book*. Wiley, 2007. [p38]

J. Fox. *An R and S-Plus Companion to Applied Regression*. Sage, 2002. [p40]

F. E. Harrell, Jr. *Regression Modeling Strategies. With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer, 2001. [p46]

I. Kreft and J. De Leeuw. *Introducing Multilevel Modeling*. Sage Publications, 1998. [p42]

I. Langford and T. Lewis. Outliers in multilevel data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 161:121–160, 1998. [p39, 40]

E. S. Lieberman. Nested analysis as a mixed-method strategy for comparative research. *American Political Science Review*, 99:435–452, 2005. [p46]

R. Nieuwenhuis, B. Pelzer, and M. te Grotenhuis. *influence.ME: Tools for detecting influential data in mixed effects models*, 2012. URL http://CRAN.R-project.org/package=influence.ME. R package version 0.9. [p38]

D. Sarkar. *Lattice. Multivariate Data Visualization with R*. Springer, 2008. [p41]

T. Snijders and J. Berkhof. Diagnostic checks for multilevel models. In J. De Leeuw and E. Meijer, editors, *Handbook of Multilevel Analysis*, chapter Diagnostic checks for multilevel models, pages 141–175. Springer, 2008. [p40]

T. Snijders and R. Bosker. *Multilevel analysis, an introduction to basic and advanced multilevel modelling*. Sage, 1999. [p38, 40]

A. Tremblay. *LMERConvenienceFunctions: A suite of functions to back-fit fixed effects and forward-fit random effects, as well as other miscellaneous functions.*, 2012. URL http://CRAN.R-project.org/package=LMERConvenienceFunctions. R package version 1.6.8.2. [p39]

T. Van der Meer, M. te Grotenhuis, and B. Pelzer. Influential cases in multilevel modeling. A methodological comment. *American Sociological Review*, 75:173–178, 2010. [p38, 40, 46]

Rense Nieuwenhuis
Institute for Innovation and Governance Studies (IGS), University of Twente
P.O. Box 217, 7500 AE, Enschede
The Netherlands
r.nieuwenhuis@utwente.nl

Manfred te Grotenhuis
Radboud University
Nijmegen
The Netherlands
m.tegrotenhuis@maw.ru.nl

Ben Pelzer
Radboud University
Nijmegen
The Netherlands
b.pelzer@maw.ru.nl

The crs Package: Nonparametric Regression Splines for Continuous and Categorical Predictors

by Zhenghua Nie and Jeffrey S Racine

Abstract A new package **crs** is introduced for computing nonparametric regression (and quantile) splines in the presence of both continuous and categorical predictors. B-splines are employed in the regression model for the continuous predictors and kernel weighting is employed for the categorical predictors. We also develop a simple R interface to NOMAD, which is a mixed integer optimization solver used to compute optimal regression spline solutions.

Introduction

Regression splines constitute a popular approach for the nonparametric estimation of regression functions though they are restricted to continuous predictors as are smoothing splines (see e.g. `smooth.spline` which is limited to one numeric predictor) and locally weighted scatterplot smoothing (see `loess` which is limited to four numeric predictors). However, it is not uncommon to encounter relationships involving a mix of numeric and categorical predictors, and the **crs** package implements a regression spline framework for accomplishing this task.

The **crs** package implements the approaches described in Ma et al. (2012) and Ma and Racine (2012) when the option `kernel=TRUE` is selected (default) as described below. The categorical predictors can be handled in two ways, (i) using kernel weighting where the kernel functions are tailored to the discrete support of the categorical predictors (Racine and Li, 2004), and (ii) using indicator basis functions. The fundamental difference between these two approaches is that the use of indicator basis functions consumes degrees of freedom via the number of columns in the basis matrix, while kernel weighting does not. As well, this package implements a range of related methods and provides options that we hope will make it appealing for applied projects, research, and pedagogical purposes alike.

Both semiparametric additive models and fully nonparametric models are implemented. Data-driven methods can be used for selecting the spline degree, number of segments/knots, and bandwidths: leave-one-out cross-validation (`cv.func = "cv.ls"`) (Stone, 1974, 1977), generalized cross-validation (`cv.func="cv.gcv"`) (Craven and Wahba, 1979), and the information-based criterion (`cv.func="cv.aic"`) proposed by Hurvich et al. (1998). Derivatives of arbitrary order can readily be constructed. One of the computational challenges encountered is to obtain the optimal spline degree (non-negative integer), number of segments/knots (positive integer), and bandwidth (bounded and real-valued) for each predictor. We overcome this challenge by providing an interface to NOMAD (Abramson et al., 2011; Le Digabel, 2011).

Before proceeding we briefly provide an overview of some important implementation details:

1. The degree of the spline and number of segments (i.e. knots minus one) for each continuous predictor can be set manually as can the bandwidths for each categorical predictor (if appropriate).

2. Alternatively, any of the data-driven criteria (i.e. `cv.func="..."`) could be used to select either the degree of the spline (holding the number of segments/knots fixed at any user-set value) and bandwidths for the categorical predictors (if appropriate), or the number of segments (holding the degree of the spline fixed at any user-set value) and bandwidths for the categorical predictors (if appropriate), or the number of segments and the degree of the spline for each continuous predictor and bandwidths for each categorical predictor (if appropriate).

3. When indicator basis functions are used instead of kernel smoothing, whether to include each categorical predictor or not can be specified manually or chosen via any `cv.func` method.

4. We allow the degree of the spline for each continuous predictor to include zero, the inclusion indicator for each categorical predictor to equal zero, and the bandwidth for each categorical predictor to equal one, and when the degree/inclusion indicator is zero or the bandwidth is one, the variable is thereby removed from the regression: in this manner, irrelevant predictors can be automatically removed by any `cv.func` method negating the need for pre-testing (mirroring findings detailed in Hall et al. (2004, 2007) for kernel regression).

The design philosophy underlying the **crs** package aims to closely mimic the behaviour of the `lm` function. Indeed, the implementation relies on `lm` and its supporting functions for computation of the

spline coefficients, delete-one residuals, fitted values, prediction and the like. 95% confidence bounds for the fit and derivatives are constructed from asymptotic formulae and automatically generated. Below we describe in more detail the specifics of the implementation for the interested reader. Others may wish to jump to the illustrative example that appears towards the end of this article or simply install and load the package and run `example(crs)`.

Differences between existing spline methods and those in the crs package

Readers familiar with existing R-functions and packages for spline modelling will naturally be wondering what the difference is between the **crs** package and existing spline-based procedures such as

- `smooth.spline` in R base
- `spm` in the **SemiPar** package (Wand, 2010)
- `gam` in the **mgcv** package (Wood, 2006)
- `ssanova` in the **gss** package (Gu, 2012)
- `gam` in the **gam** package (Hastie, 2011)

First we note that the functions `smooth.spline` and `ssanova` are based on smoothing spline methodology, while `spm` uses penalized splines but `gam` in the **gam**/**mgcv** packages allows for smoothing splines, penalized splines, and regression splines. The **crs** package is restricted to 'regression splines' which differs in a number of ways from 'smoothing splines' and 'penalized splines', the fundamental difference being that smoothing/penalized splines use the data points themselves as potential knots and penalize 'roughness' (typically the second derivative of the estimate) while regression splines place knots at equidistant/equiquantile points and do not explicitly penalize roughness, rather, they rely on various cross-validatory approaches for model selection. We direct the interested reader to Wahba (1990) for a treatment of smoothing splines. The **crs** package is one of the few packages devoted to regression spline techniques. We also provide quantile regression splines via the option `tau=`τ ($\tau \in (0,1)$).

Second, many of the aforementioned smoothing spline implementations are semiparametric in nature, the semiparametric additive model being particularly popular. Though semiparametric models exist to circumvent the curse of dimensionality, it does not come without cost. That is, the burden of determining whether semiparametric or nonparametric approaches would be warranted in any given situation is placed squarely on the researcher's shoulder. Unlike many existing spline methods in R, the implementation in the **crs** package is designed so that every parameter that must be chosen by the researcher can be data-driven (via cross-validation) so that such choices adapt to the data at hand *including* whether to use a semiparametric or nonparametric model. This is accomplished using options such as `basis="auto"`, `knots="auto"`, and `complexity="degree-knots"` (`basis="auto"` deciding whether to use an additive basis or tensor product basis in multivariate settings, `knots="auto"` whether to use equispaced knots or quantile knots, and `complexity="degree-knots"` determining both the spline degrees and number of knots).

Generally speaking, almost all of these existing smoothing spline approaches can handle mixed datatypes though their approaches to the treatment of categorical variables often differ (none use categorical kernel smoothing as in the **crs** package).

The underlying model

Regression spline methods can be limited in their potential applicability as they are based on *continuous* predictors. However, in applied settings we often encounter *categorical* predictors such as strength of preference ("strongly prefer", "weakly prefer", "indifferent") and so forth.

We wish to model the unknown conditional mean in the following location-scale model,

$$Y = g(\mathbf{X}, \mathbf{Z}) + \sigma(\mathbf{X}, \mathbf{Z})\,\varepsilon,$$

where $g(\cdot)$ is an unknown function, $\mathbf{X} = (X_1, \ldots, X_q)^{\mathrm{T}}$ is a q-dimensional vector of continuous predictors, $\mathbf{Z} = (Z_1, \ldots, Z_r)^{\mathrm{T}}$ is an r-dimensional vector of categorical predictors, and $\sigma^2(\mathbf{X}, \mathbf{Z})$ is the conditional variance of Y given $\mathbf{X}$ and $\mathbf{Z}$. Letting $\mathbf{z} = (z_s)_{s=1}^{r}$, we assume that z_s takes c_s different values in $D_s \equiv \{0, 1, \ldots, c_s - 1\}$, $s = 1, \ldots, r$, and c_s is a finite positive constant.

For the continuous predictors the regression spline model employs the B-spline routines in the GNU Scientific Library (Galassi et al., 2009). The B-spline function is the maximally differentiable interpolative basis function (B-spline stands for 'basis-spline').

Heuristically, we conduct linear (in parameter) regression using the R function `lm`, however, we replace the continuous predictors with B-splines of potentially differing order and number of segments for each continuous predictor. For the tensor product bases we set `intercept=TRUE` for each univariate spline basis, while for the additive spline bases we adopt the `intercept=FALSE` variants and include an intercept term in the model (the B-splines will therefore not sum to one, i.e. an order m B-spline with one segment (two knots/breakpoints) has $m+1$ columns and we drop the first as is often done, though see Zhou and Wolfe (2000) for an alternative approach based upon shrinkage methods). This allows multiple bases to coexist when there is more than one continuous predictor. The tensor product basis is given by

$$B_1 \otimes B_2 \otimes \cdots \otimes B_p,$$

where $\otimes$ is the Kronecker product where the products operate column-wise and B_j is the basis matrix for predictor j as outlined above. We also support a 'generalized' polynomial B-spline basis that consists of a varying-order polynomial with appropriate interactions. When additive B-spline bases are employed we have a semiparametric 'additive' spline model (no interaction among variables unless explicitly specified). When the tensor product or generalized polynomial is employed we have a fully nonparametric model (interaction among all variables). Whether to use the additive, tensor product, or generalized polynomial bases can be pre-specified or automatically determined via any `cv.func` method (see the options for `basis=` in `?crs`).

We offer the default option to use categorical kernel weighting (`lm(...,weights=L)`) to handle the presence of categorical predictors (see below for a description of `L`). We also offer the option of using indicator basis functions for the categorical predictors.

Weighted least-squares estimation of the underlying model

The unknown function $g(\mathbf{X},\mathbf{Z})$ can be approximated by $\mathcal{B}(\mathbf{X})^{\mathrm{T}}\beta(\mathbf{Z})$, where $\beta(\mathbf{Z})$ is a vector of coefficients and $\mathcal{B}(\mathbf{X})$ the B-spline basis matrix described above.

We estimate $\beta(\mathbf{z})$ by minimizing the following weighted least squares criterion,

$$\widehat{\beta}(\mathbf{z}) = \underset{\beta(\mathbf{z})\in\mathbb{R}^{\mathbf{K}_n}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\{ Y_i - \mathcal{B}(\mathbf{X}_i)^{\mathrm{T}}\beta(\mathbf{z}) \right\}^2 L(\mathbf{Z}_i,\mathbf{z},\lambda).$$

Placement of knots

The user can determine where knots are to be placed using one of two methods:

1. knots can be placed at equally spaced quantiles whereby an equal number of observations lie in each segment ('quantile knots').
2. knots can be placed at equally spaced intervals ('uniform knots').

If preferred, this can be determined automatically using the option `knots="auto"`.

Kernel weighting

Let Z_i be an r-dimensional vector of categorical/discrete predictors. We use z_s to denote the s-th component of z, we assume that z_s takes c_s different values in $D_s \equiv \{0,1,\ldots,c_s-1\}$, $s=1,\ldots,r$, and let $c_s \geq 2$ be a finite positive constant.

For an unordered categorical predictor, we use a variant of the kernel function outlined in Aitchison and Aitken (1976) defined as

$$l(Z_{is},z_s,\lambda_s) = \begin{cases} 1, & \text{when } Z_{is}=z_s, \\ \lambda_s, & \text{otherwise.} \end{cases} \qquad (1)$$

Let $\mathbf{1}(A)$ denote the usual indicator function, which assumes the value one if A holds true, zero otherwise. Using (1), we can construct a product kernel weight function given by

$$L(\mathbf{Z}_i,\mathbf{z},\lambda) = \prod_{s=1}^{r} l(Z_{is},z_s,\lambda_s) = \prod_{s=1}^{r} \lambda_s^{\mathbf{1}(Z_{is}\neq z_s)},$$

while for an ordered categorical we use the function defined by

$$\tilde{l}(Z_{is},z_s,\lambda_s) = \lambda_s^{|Z_{is}-z_s|}$$

and modify the product kernel function appropriately. When Z contains a mix of ordered and unordered variables we select the appropriate kernel for each variable's type when constructing the product kernel $L(\mathbf{Z}_i,\mathbf{z},\lambda)$.

Note that when $\lambda_s = 1$ all observations are 'pooled' over categorical predictor s hence the variable z_s is removed from the resulting estimate, while when $\lambda_s = 0$ only observations lying in a given cell are used to form the estimate.

Additional estimation details

Estimating the model requires construction of the spline bases and their tensor product (if specified) along with the categorical kernel weighting function. Then, for a given degree and number of segments for each continuous predictor and bandwidth for each categorical predictor (or indicator bases if `kernel=FALSE`), the model is fit via least-squares.

All smoothing parameters can be set manually by the user if so desired, however be forewarned that you must use the option `cv="none"` otherwise the values specified manually will become the starting points for search when `cv="nomad"` ('nonsmooth mesh adaptive direct search (NOMAD)', see Abramson et al. (2011) and Le Digabel (2011)). Currently, we provide a simple R interface to NOMAD (see the section below) in the **crs** package which also can be applied to solve other mixed integer optimization problems.

The degree, segment, and bandwidth vectors can be jointly determined via any `cv.func` method by setting the option `cv="nomad"` or `cv="exhaustive"` (exhaustive search). Here we have to solve nonlinear non-convex and non-differentiable mixed integer constrained optimization problems to obtain the optimal degree, number of segments, and bandwidth for each predictor.

Setting the option `cv="nomad"` (default) computes NOMAD-based cross-validation directed search while setting `cv="exhaustive"` computes exhaustive cross-validation directed search for each unique combination of the degree and segment vector for each continuous predictor from `degree=degree.min` through `degree=degree.max` (default 0 and 10, respectively) and from `segments=segments.min`

through `segments=segments.max` (default 1 and 10, respectively).

When `kernel=TRUE` (default) setting the option `cv="exhaustive"` computes bandwidths ($\in [0,1]$) obtained via numerical minimization (using `optim`) for each unique combination of the degree and segment vectors (restarting can be conducted via `restarts=`). When setting `cv="nomad"` the number of multiple starts can be controlled by `nmulti=` (default is 5). The model possessing the lowest criterion function value over the `nmulti` restarts is then selected as the final model.

Note that `cv="exhaustive"` is often unfeasible (this combinatoric problem can become impossibly large to compute in finite time) hence `cv="nomad"` is the default. However, with `cv="nomad"` one should set `nmulti=` to some sensible value greater than zero to avoid becoming trapped in local minima (default `nmulti=5`).

Data-driven smoothing parameter criteria

We incorporate three popular approaches for setting the smoothing parameters of the regression spline model, namely least-squares cross-validation, generalized cross-validation, and an AIC method corrected for use in nonparametric settings.

Let the fitted value from the regression spline model be denoted $\hat{Y}_i = B_m(X_i)^T\hat{\beta}(Z_i)$. Letting $\hat{\varepsilon}_i = Y_i - \hat{Y}_i$ denote the ith residual from the categorical regression spline model, the least-squares cross-validation function is given by

$$CV = \frac{1}{n}\sum_{i=1}^{n}\frac{\hat{\varepsilon}_i^2}{(1-h_{ii})^2}$$

and this computation can be done with effectively one pass through the data set, where h_{ii} denotes the ith diagonal element of the spline basis projection matrix (see below for details). Since h_{ii} is computed routinely for robust diagnostics in R, this can be computed along with (and hence as cheaply as) the vector of spline coefficients themselves. Thus least-squares cross-validation is computationally appealing, particularly for large data sets.

Let H denote the $n \times n$ weighting matrix such that $\hat{Y} = HY$ with its ith diagonal element given by H_{ii} where $\mathrm{tr}(H)$ means the trace of H which is equal to $\sum_{i=1}^{n} h_{ii}$. The matrix H is often called the 'hat matrix' or 'smoother matrix' and depends on X but not on Y. The 'generalized' cross-validation function is obtained by replacing h_{ii} in the above formula with its average value denoted $\mathrm{tr}(H)/n$ (Craven and Wahba, 1979).

The information-based criterion proposed by Hurvich et al. (1998) is given by

$$\mathrm{AIC}_c = \ln(\hat{\sigma}^2) + \frac{1+\mathrm{tr}(H)/n}{1-\{\mathrm{tr}(H)+2\}/n}$$

where

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{\varepsilon}_i^2 = Y'(I-H)'(I-H)Y/n.$$

Each of these criterion functions can be minimized with respect to the unknown smoothing parameters either by numerical optimization procedures or by exhaustive search.

Though each of the above criteria are asymptotically equivalent in terms of the bandwidths they deliver ($\mathrm{tr}(H)/n \to 0$ as $n \to \infty$), they may differ in finite-sample settings for a small smoothing parameter (large $\mathrm{tr}(H)/n$) with the AIC_c criterion penalizing more heavily when undersmoothing than either the least-squares cross-validation or generalized cross-validation criteria (the AIC_c criterion effectively applies an infinite penalty for $\mathrm{tr}(H)/n \geq 1/2$).

Pruning

Once a model has been selected via cross-validation (i.e. `degree`, `segments`, `include` or `lambda` have been selected), there is the opportunity to (potentially) further refine the model by adding the option `prune=TRUE` to the `crs` function call. Pruning is accomplished by conducting stepwise cross-validated variable selection using a modified version of the `stepAIC` function in the R **MASS** package where the function `extractAIC` is replaced with the function `extractCV` with additional modifications where necessary. Pruning of potentially superfluous bases is undertaken, however, the pruned model (potentially containing a subset of the bases) is returned *only if its cross-validation score improves upon the model being pruned*. When this is not the case a warning is issued to this effect. A final pruning stage is commonplace in spline frameworks and may positively impact on the finite-sample efficiency of the resulting estimator depending on the rank of the model being pruned. Note that this option can only be applied when `kernel=FALSE` (or when there exist only numeric predictors).

A simple R interface to NOMAD

The **crs** package has included a simple R interface to the NOMAD optimization solver called `snomadr`. `snomadr` implements the NOMAD library which is an open source C++ implementation of the Mesh Adaptive Direct Search (MADS) algorithm designed for constrained optimization of blackbox functions (Abramson et al., 2011; Le Digabel, 2011). `snomadr` can be seen as a standalone interface to this optimization solver, though we would like to point out that the authors of `NOMAD` are currently working on an R package for NOMAD that we expect to be much more comprehensive than the simple interface provided here. The principle behind developing our in-

terface is simply to shield the user from setting up the optimization model underlying our regression spline implementation. For what follows we will not discuss the performance of the solver NOMAD, rather we direct the interested reader to Abramson et al. (2011); Le Digabel (2011) and the references therein.

The structure of the `snomadr` interface is similar to the interface in the R package **ipoptr** (`http://www.ucl.ac.uk/~uctpjyy/ipoptr.html`) which exports the R interface to the optimization solver IPOPT (Wächter and Biegler, 2006). The interface is split into two major portions, one is the R code called 'snomadr.R' and another is a C++ program called 'snomadr.cpp'. The role of 'snomadr.R' is to set up the optimization model, define options and then call the C++ code to solve the optimization problem. 'snomadr.cpp' will receive arguments from the R code and then call the NOMAD library functions to solve the problem and return results to 'snomadr.R'. `.Call` is used to transfer arguments and results between the R and C++ languages. For further details see `?snomadr`.

Illustrative example

By way of illustration we consider a simple simulated example involving one continuous and one categorical predictor.

```
set.seed(42)
n <- 1000
x <- runif(n)
z <- rbinom(n, 1, .5)
y <- cos(2 * pi * x) + z + rnorm(n, sd=0.25)
z <- factor(z)
model <- crs(y ~ x + z)
summary(model)

Call:
crs.formula(formula = y ~ x + z)

Kernel Weighting/B-spline Bases Regression
   Spline

There is 1 continuous predictor
There is 1 categorical predictor
Spline degree/number of segments for x: 3/4
Bandwidth for z: 0.0008551836
Model complexity proxy: degree-knots
Knot type: quantiles
Training observations: 1000
Trace of smoother matrix: 14
Residual standard error: 0.2453 on 993
    degrees of freedom
Multiple R-squared: 0.927,
    Adjusted R-squared: 0.9265
F-statistic: 962.9 on 13 and 986 DF,
    p-value: 0
Cross-validation score: 0.061491531
Number of multistarts: 5
```

The function `crs()` called in this example returns a `"crs"` object. The generic functions `fitted` and `residuals` extract (or generate) estimated values and residuals. Furthermore, the functions `summary`, `predict`, and `plot` (options `mean=FALSE`, `deriv=`*i*, `ci=FALSE`, `plot.behavior = c("plot", "plot-data", "data")`, where *i* is a positive integer) support objects of this type.

Figure 1 presents summary output in the form of partial regression surfaces (predictors not appearing on the axes are held constant at their medians/modes). Note that for this simple example we used the option `plot(model,mean=TRUE)` which presents 'partial regression plots'.[1]

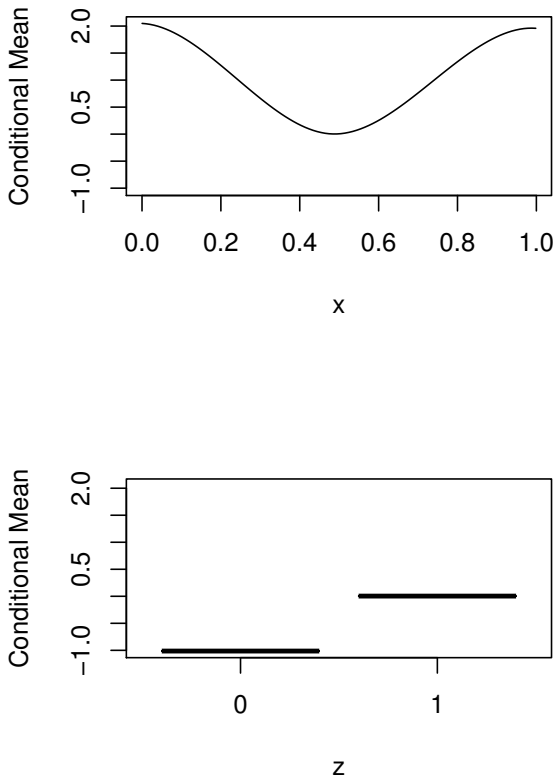


Figure 1: The partial regression plot for one categorical and one continuous predictor.

Next we plot the first partial derivative (which plots the partial derivative with respect to the continuous predictor holding the categorical predictor at its modal value and next the difference between the regression function when the categorical predictor equals 0 and 1 holding the continuous predictor

[1] A 'partial regression plot' is simply a 2D plot of the outcome y versus one predictor x_j when all other predictors are held constant at their respective medians/modes (this can be changed by the user).

at its median) for this model in Figure 2. Note that here we used the option `plot(model,deriv=1)`.

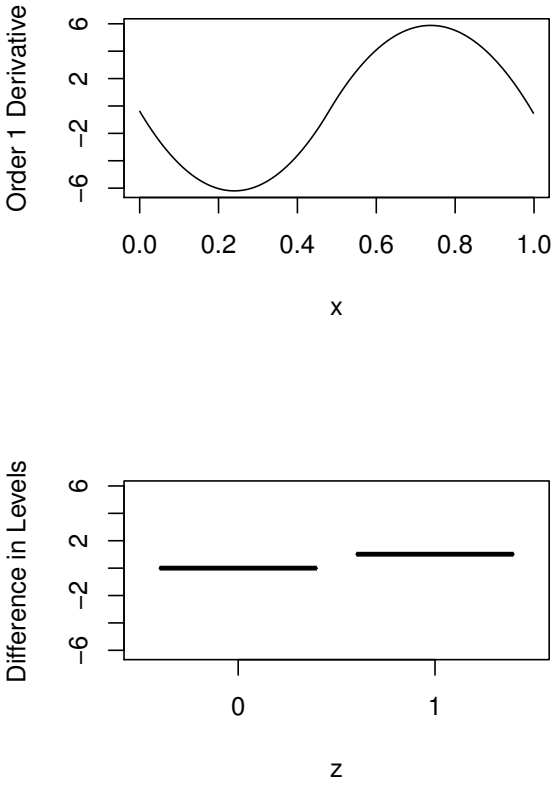


Figure 2: The partial gradient plot for one categorical and one continuous predictor.

Comparison with existing spline methods

We now compare and contrast some existing spline implementations in R with the regression spline implementation contained in the **crs** package via two illustrative Monte Carlo simulations.

The bivariate one continuous predictor case

We first consider the simple and popular case of nonparametrically modeling the relationship between a response Y and univariate predictor X. Here the purpose is to help the reader assess the performance of regression splines versus smoothing splines. We consider four data generating processes (DGPs), the sine and cosine functions, the absolute value function, and the Doppler function given by

$$g(x) = \sqrt{x(1-x)} \times \sin\left(\frac{2\pi(1+2^{-7/5})}{x+2^{-7/5}}\right)$$

For the sine, cosine, and Doppler functions X is $U[0,1]$, while for the absolute value function X is $U[-1,1]$. We vary the signal/noise ratio by standardizing the DGP to have mean zero and unit variance and setting σ for the Gaussian noise to $1/4, 1/2, 1, 2$ which produces expected in-sample fits of $R^2 = 0.95, 0.80, 0.50, 0.20$ for the oracle estimator (i.e. one that uses knowledge of the DGP).

We compare `crs`-based regression splines with smoothing spline methods `gam` in the **mgcv** package, `spm` in the **SemiPar** package, `ssanova` in the **gss** package, and `loess` in base R. All methods in this first simulation use default settings.

We draw $M = 1{,}000$ replications of size $n = 1{,}000$ from each DGP, compute the mean square error (MSE) for each model, and report the median value of each model's MSE relative to that for `crs` over the $M = 1{,}000$ replications in Table 1 below (numbers less than one indicate that a method is more efficient than `crs`).

Table 1: MSE efficiency of the various methods relative to `crs` (numbers less than one indicate more efficient than `crs`).

$\sin(2\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	0.97	1.24	1.02	8.06
$\sigma = 0.50$	0.80	0.89	0.81	2.15
$\sigma = 1.00$	0.78	0.78	0.78	0.92
$\sigma = 2.00$	0.84	0.79	0.83	0.71
$\sin(4\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	0.89	1.43	1.11	507.85
$\sigma = 0.50$	0.78	1.15	0.98	129.41
$\sigma = 1.00$	0.81	1.01	0.95	36.72
$\sigma = 2.00$	0.77	0.84	0.85	9.89
$\cos(2\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	1.13	1.17	1.17	39.39
$\sigma = 0.50$	0.99	1.00	1.04	11.18
$\sigma = 1.00$	0.98	0.93	1.00	3.67
$\sigma = 2.00$	0.89	0.82	0.91	1.38
$\cos(4\pi x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	2.78	1.54	1.30	299.88
$\sigma = 0.50$	1.31	1.25	1.12	79.85
$\sigma = 1.00$	0.93	1.02	0.97	21.42
$\sigma = 2.00$	0.82	0.89	0.87	6.09
$\text{abs}(x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	1.55	0.91	0.98	14.59
$\sigma = 0.50$	1.20	1.05	1.11	5.97
$\sigma = 1.00$	1.23	1.18	1.26	2.73
$\sigma = 2.00$	1.41	1.36	1.43	1.61
$\text{doppler}(x)$	gam	spm	ssanova	loess
$\sigma = 0.25$	65.45	1.67	1.68	359.63
$\sigma = 0.50$	18.53	1.39	1.42	99.51
$\sigma = 1.00$	5.15	1.14	1.19	25.40
$\sigma = 2.00$	1.78	1.03	1.04	6.81

Table 1 reveals some general patterns. For smooth low frequency DGPs such as the sine and cosine, `gam`, `spm`, and `ssanova` turn in similar performances depending on the signal/noise ratio. However, `loess` does not do well with these DGPs. For the absolute value and Doppler DGPs `gam`, `spm`, `ssanova` and `loess` on balance perform worse than `crs`. From a minimax perspective, the regression spline implementation in the **crs** package performs relatively well overall. Naturally by modifying tuning parameters one could expect to improve the performance of the smoothing spline methods (this would apply to `crs` as well as default settings are used throughout). The point to be made is simply that the regression spline implementation in **crs** appears to adapt fairly well to the underlying DGP and may therefore be of interest to readers.

The multivariate case, mixed predictor types

Next we consider mixed predictor multivariate DGPs with additive and multiplicative features. The additive DGP is of the form

```
x1 <- runif(n)
x2 <- runif(n)
z <- rbinom(n, 1, .5)
dgp <- cos(2*pi*x1) + sin(2*pi*x2) + z
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

while the multiplicative DGPs are of the form

```
x1 <- runif(n)
x2 <- runif(n)
z <- rbinom(n, 1, .5)
dgp <- cos(2*pi*x1) * sin(2*pi*x2) * z
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

and the radial function given by

```
x1 <- runif(n, -5, 5)
x2 <- runif(n, -5, 5)
z <- rbinom(n, 1, .5)
dgp <- sin(sqrt(x1^2 + x2^2 + z))/
       sqrt(x1^2 + x2^2 + z)
dgp <- (dgp - mean(dgp))/sd(dgp)
y <- dgp + rnorm(n, sd=sigma)
z <- factor(z)
```

We again vary the signal/noise ratio by standardizing the DGP and setting σ for the Gaussian noise to $1/4, 1/2, 1, 2$ which produces expected in-sample fits of $R^2 = 0.95, 0.80, 0.50, 0.20$ for the oracle estimator (i.e. one that uses knowledge of the DGP). We estimate the following eight models,

```
model <- crs(y ~ x1 + x2 + z)
# gam (mgcv) additive
model <- gam(y ~ s(x1) + s(x2) + z)
# gam (mgcv) tensor to admit interactions
model <- gam(y ~ t2(x1, x2, k=k) + z)
# gam (mgcv) tensor with smooth for every z
model <- gam(y ~ t2(x1, x2, k=k, by=z) + z)
# spm (SemiPar) additive
model <- spm(y ~ f(x1) + f(x2) + z)
# spm (SemiPar) tensor product
model <- spm(y ~ f(x1, x2) + z)
# ssanova (gss) additive
model <- ssanova(y ~ x1 + x2 + z)
# ssanova (gss) tensor product
model <- ssanova(y ~ x1 * x2 + z)
```

We draw $M = 1000$ replications of size $n = 1000$ from each DGP, compute the MSE for each model, and report the median value of each model's MSE relative to that for `crs` over the M replications in Table 2 below.

Table 2: MSE efficiency of the various methods relative to `crs` (numbers less than one indicate more efficient than `crs`).

Additive DGP ($\cos(2\pi x_1) + \sin(2\pi x_2) + z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	0.60	1.57	2.11	0.64	1.71	0.62	0.70
0.50	0.57	0.94	1.56	0.57	1.24	0.57	0.65
1.00	0.55	0.83	1.44	0.51	0.95	0.54	0.65
2.00	0.52	0.75	1.35	0.49	0.72	0.51	0.60

Additive DGP ($\cos(4\pi x_1) + \sin(4\pi x_2) + z$))

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	0.72	1.07	1.88	0.77	5.76	0.65	0.72
0.50	0.54	0.94	1.70	0.63	2.50	0.58	0.65
1.00	0.52	0.90	1.65	0.57	1.53	0.57	0.62
2.00	0.53	0.90	1.64	0.56	1.17	0.55	0.63

Multiplicative DGP ($\cos(2\pi x_1) \times \sin(2\pi x_2) \times z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	228.82	111.54	0.78	229.05	111.41	229.17	112.97
0.50	95.18	46.93	1.09	95.26	46.81	95.31	47.46
1.00	30.23	15.50	1.16	30.25	15.55	30.25	15.60
2.00	9.61	5.39	1.14	9.60	5.49	9.59	5.32

Multiplicative DGP ($\cos(4\pi x_1) \times \sin(4\pi x_2) \times z$)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	93.39	44.36	0.62	93.52	51.31	93.58	54.14
0.50	30.05	14.66	0.64	30.09	16.85	30.11	17.71
1.00	9.92	5.28	0.74	9.93	5.92	9.94	6.56
2.00	3.40	2.26	0.86	3.39	2.50	3.39	3.37

Multiplicative DGP (radial)

		gam			spm		ssanova	
σ	(add)	(int)	(by)	(add)	(te)	(add)	(te)
0.25	89.66	2.18	1.16	89.07	2.29	89.60	2.21
0.50	31.35	1.27	1.30	31.18	1.29	31.35	1.21
1.00	12.65	1.19	1.72	12.56	1.09	12.68	1.08
2.00	4.49	1.10	1.82	4.44	0.88	4.51	0.99

For the **mgcv** routine `gam`, a referee noted that `?choose.k` could be consulted where it suggests using `gam.check()` to help guide the appropriate number of knots and suggested non-stochastic values of `k=5` for the additive DGP with 2π and `k=10` for the remaining DGPs (these values were used in Table 2

for gam via the argument k=k rather than the default values, while crs uses stochastic values for the basis and all smoothing parameters).

When the DGP is additive, it is clear that the smoothing splines that presume additivity are more efficient than crs for this DGP. Note that this is an oracle type of property in the sense that the user does not in general have knowledge of the true DGP, but if they did and used this information their model would naturally be more efficient than a method that did not (we elaborate further on this below). The regression spline implementation in the **crs** package does not presume this knowledge when basis="auto" is used (the default) though it allows for the possibility. However, were one to use the incorrect smoothing spline model when the DGP is additive (e.g. assume interaction when it is not present), one could do worse than crs as can be seen in Table 2. But note that when the DGP is non-additive, crs appears to be more efficient than the smoothing spline approaches even when they employ, say, tensor basis functions to model the non-additive relationship and are adapted to admit the presence of the binary factor predictor (except gam which, as noted above, uses a number of knots k that is optimized for these DGPs, rather than the defaults which were used for all other methods). From a minimax perspective, the regression spline implementation in the **crs** package performs relatively well overall. Apart from selecting the number of knots for gam as described above, default settings were used throughout and therefore it is likely that efficiency could be improved in some cases by further tuning (this would apply to crs as well, naturally).

To elaborate further on the issue of tuning and efficiency, if instead of using defaults for crs we were to assume additivity as done for its peers in columns 2, 5, and 7 in Table 2 (options basis="additive", kernel=FALSE), then for the additive DGP in Table 2 the first row would instead be 0.94, 2.43, 3.29, 1.00, 2.69, 0.98, and 1.08, and we observe that the efficiency of crs improves substantially even when using a stochastic choice of the B-spline degree and number of knots, while the other methods use non-stochastic choices for the number of knots that are constant over all $M = 1{,}000$ Monte Carlo replications. And if we followed the lead of the anonymous referee who kindly suggested using non-stochastic values for the number of knots (i.e. k=5) for gam in **mgcv** based on replications from the additive DGP used for row 1 of Table 2, we might choose non-stochastic values degree=c(4,4) and segments=c(3,3) based on cross-validation and the first row of Table 2 would instead be 1.18, 3.09, 4.19, 1.25, 3.42, 1.22, and 1.36. Furthermore, if in addition we used the option prune=TRUE enabling post-estimation pruning of the model, the first row of Table 2 would instead be 1.48, 3.91, 5.35, 1.59, 4.28, 1.53, and 1.72 (note that this can only be used in conjunction with the option kernel=FALSE). The point to be made is that indeed efficiency can be improved in some cases by tuning of smoothing parameters and choice of the basis, and this also holds for crs itself.

Demos, data, and additional information

There exist a range of demonstrations available via demo() including

1. radial_persp: R code for fitting and plotting a 3D perspective plot for the 'radial' function.
2. radial_rgl: R code for fitting and generating a 3D real-time rendering plot for the 'radial' function using OpenGL (requires the **rgl** package (Adler and Murdoch, 2012)).
3. sine_rgl: R code for fitting and generating a 3D real-time rendering plot for a product sine function using OpenGL.
4. radial_constrained_mean: R code for constrained radial function estimation.
5. cqrs.R: R code for fitting and plotting quantile regression splines.

There exist a number of datasets including

1. cps71: Canadian cross-section wage data consisting of a random sample taken from the 1971 Canadian Census Public Use Tapes for male individuals having common education (grade 13). There are 205 observations in total (contributed by Aman Ullah).
2. Engel95: British cross-section data consisting of a random sample taken from the British Family Expenditure Survey for 1995. The households consist of married couples with an employed head-of-household between the ages of 25 and 55 years. There are 1655 household-level observations in total (contributed by Richard Blundell).
3. wage1: Cross-section wage data consisting of a random sample taken from the U.S. Current Population Survey for the year 1976. There are 526 observations in total (contributed by Jeffrey Wooldridge).

We have tried to overload the crs method and associated S3 plot method to accommodate the needs of a range of users and to automate a number of routine tasks. Warning messages are produced where possible to guide the user towards the most appropriate choice of settings. User specified weights can be provided to allow for weighted least squares and weighted cross-validation. In addition, we have a function crsiv that implements instrumental variable regression splines that may be of interest to

some users (see `?crsiv` for details). Furthermore, quantile regression splines are supported by setting `tau=` a scalar in the range $(0,1)$. Finally, we would like to direct the reader to the vignettes `crs`, `crs_faq`, and `spline_primer` for further examples, frequently asked questions, and background information.

Acknowledgements

Racine would like to gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada (http://www.nserc.ca), the Social Sciences and Humanities Research Council of Canada (http://www.sshrc.ca), and the Shared Hierarchical Academic Research Computing Network (http://www.sharcnet.ca). We would also like to gratefully acknowledge Sébastien Le Digabel and members of the NOMAD project for their contributions to the FOSS community. Finally, we are indebted to the editors and two anonymous referees for their invaluable comments and suggestions.

Bibliography

M. Abramson, C. Audet, G. Couture, J. Dennis Jr., and S. Le Digabel. The NOMAD project, 2011. URL http://www.gerad.ca/nomad. [p48, 50, 51, 52]

D. Adler and D. Murdoch. *rgl: 3D visualization device system (OpenGL)*, 2012. URL http://CRAN.R-project.org/package=rgl. R package version 0.92.892. [p55]

J. Aitchison and C. G. G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976. [p50]

P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 13:377–403, 1979. [p48, 51]

M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual*, third edition, January 2009. URL http://http://www.gnu.org/software/gsl/. Version 1.12. [p49]

C. Gu. *gss: General Smoothing Splines*, 2012. URL http://CRAN.R-project.org/package=gss. R package version 2.0-9. [p49]

P. Hall, J. S. Racine, and Q. Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004. [p48]

P. Hall, Q. Li, and J. S. Racine. Nonparametric estimation of regression functions in the presence of irrelevant regressors. *The Review of Economics and Statistics*, 89:784–789, 2007. [p48]

T. Hastie. *gam: Generalized Additive Models*, 2011. URL http://CRAN.R-project.org/package=gam. R package version 1.06.2. [p49]

C. M. Hurvich, J. S. Simonoff, and C. L. Tsai. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. *Journal of the Royal Statistical Society Series B*, 60:271–293, 1998. [p48, 51]

S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):44:1–44:15, 2011. [p48, 50, 51, 52]

S. Ma and J. S. Racine. Additive regression splines with irrelevant categorical and continuous regressors. *Statistica Sinica*, 2012. In Press. [p48]

S. Ma, J. S. Racine, and L. Yang. Spline regression in the presence of categorical predictors. *Journal of Multivariate Analysis*, 2012. Revised and Resubmitted. [p48]

J. S. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004. [p48]

C. J. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society*, 36:111–147, 1974. [p48]

C. J. Stone. Consistent nonparametric regression. *Annals of Statistics*, 5:595–645, 1977. [p48]

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. URL http://projects.coin-or.org/ipopt. [p52]

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990. [p49]

M. Wand. *SemiPar: Semiparametic Regression*, 2010. URL http://CRAN.R-project.org/package=SemiPar. R package version 1.0-3. [p49]

S. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006. [p49]

S. Zhou and D. A. Wolfe. On derivative estimation in spline regression. *Statistica Sinica*, 10:93–108, 2000. [p49]

Jeffrey S. Racine, Zhenghua Nie
McMaster University
1280 Main Street West, Hamilton, Ontario
Canada
racinej@mcmaster.ca
niez@mcmaster.ca

Rfit: Rank-based Estimation for Linear Models

by John D. Kloke and Joseph W. McKean

Abstract In the nineteen seventies, Jurečková and Jaeckel proposed rank estimation for linear models. Since that time, several authors have developed inference and diagnostic methods for these estimators. These rank-based estimators and their associated inference are highly efficient and are robust to outliers in response space. The methods include estimation of standard errors, tests of general linear hypotheses, confidence intervals, diagnostic procedures including studentized residuals, and measures of influential cases. We have developed an R package, **Rfit**, for computing of these robust procedures. In this paper we highlight the main features of the package. The package uses standard linear model syntax and includes many of the main inference and diagnostic functions.

Introduction

Rank-based estimators were developed as a robust, nonparametric alternative to traditional likelihood or least squares estimators. Rank-based regression was first introduced by Jurečková (1971) and Jaeckel (1972). McKean and Hettmansperger (1978) developed a Newton step algorithm that led to feasible computation of these rank-based estimates. Since then a complete rank-based inference for linear models has been developed that is based on rank-based estimation analogous to the way that traditional analysis is based on least squares (LS) estimation; see Chapters 3-5 of the monograph by Hettmansperger and McKean (2011) and Chapter 9 of Hollander and Wolfe (1999). Furthermore, robust diagnostic procedures have been developed with which to ascertain quality of fit and to locate outliers in the data; see McKean and Sheather (2009) for a recent discussion. Kloke et al. (2009) extended this rank-based inference to mixed models. Thus rank-based analysis is a complete analysis analogous to the traditional LS analysis for general linear models. This rank-based analysis generalizes Wilcoxon procedures for simple location models and, further, it inherits the same high efficiency that these simple nonparametric procedures possess. In addition, weighted versions can have high breakdown (up to 50%) in factor space (Chang et al., 1999). In this paper, we discuss the **Rfit** package that we have developed for rank-based (R) estimation and inference for linear models. We illustrate its use on examples from simple regression to k-way factorial designs.

The geometry of rank-based estimation is similar to that of LS. In rank-based regression, however, we replace Euclidean distance with another measure of distance which we refer to as Jaeckel's dispersion function; see Hettmansperger and McKean (2011) for details. For a brief overview see McKean (2004).

Jaeckel's dispersion function depends on the choice of a score function. As discussed in Hettmansperger and McKean (2011), the rank-based fit and associated analysis can be optimized by a prudent choice of scores. If the form of the error distribution is known and the associated scores are used, then the the analysis is fully efficient. In **Rfit** we have included a library of score functions. The default option is to use Wilcoxon (linear) scores, however it is straightforward to create user-defined score functions. We discuss score functions further in a later section.

Others have developed software for rank-based estimation. Kapenga et al. (1995) developed a Fortran package and Crimin et al. (2008) developed a web interface (cgi) with Perl for this Fortran program. Terpstra and McKean (2005) developed a set of R functions to compute weighted Wilcoxon (WW) estimates including the high breakdown point rank-based (HBR) estimate proposed by Chang et al. (1999). See McKean et al. (2009) for a recent review. **Rfit** differs from the WW estimates in that its estimation algorithms are available for general scores and it uses a standard linear models interface.

The package **Rfit** allows the user to implement rank-based estimation and inference described in Chapters 3-5 of Hettmansperger and McKean (2011) and Chapter 9 of Hollander and Wolfe (1999). There are other robust packages in R. For example, the R function `rlm` of the R package **MASS** (Venables and Ripley, 2002) computes M estimates for linear models based on the ψ functions of Huber, Hampel, and Tukey (bisquare). The CRAN Task View on robust statistical methods offers robust procedures for linear and nonlinear models including methods based on M, M-S, and MM estimators. These procedures, though, do not obtain rank-based estimates and associated inference as do the procedures in **Rfit**.

Rfit uses standard linear model syntax so that those familiar with traditional parametric analysis can easily begin running robust analyses. In this paper, discussion of the assumptions are kept to a minimum and we refer the interested reader to the literature. All data sets used in demonstrating **Rfit** are included in the package.

The rest of the paper is organized as follows. The next section discusses the general linear model and rank-based fitting and associated inference. The

following section provides examples illustrating the computation of **Rfit** for linear regression. Later sections discuss **Rfit**'s computation of one-way and multi-way ANOVA as well as general scores and writing user-defined score functions for computation in **Rfit**. The final section discusses the future implementation in **Rfit** of rank-based procedures for models beyond the general linear model.

Rank-regression

As with least squares, the goal of rank-based regression is to estimate the vector of coefficients, $\boldsymbol{\beta}$, of a general linear model of the form:

$$y_i = \alpha + \boldsymbol{x}_i^T\boldsymbol{\beta} + e_i \quad \text{for } i = 1,\ldots n \tag{1}$$

where y_i is the response variable, $\boldsymbol{x}_i$ is the vector of explanatory variables, α is the intercept parameter, and e_i is the error term. We assume that the errors are iid with probability density function (pdf) $f(t)$. For convenience, we write (1) in matrix notation as follows

$$\boldsymbol{y} = \alpha\mathbf{1} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{e} \tag{2}$$

where $\boldsymbol{y} = [y_1,\ldots,y_n]^T$ is the $n \times 1$ vector of responses, $\boldsymbol{X} = [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n]^T$ is the $n \times p$ design matrix, and $\boldsymbol{e} = [e_1,\ldots,e_n]^T$ is the $n \times 1$ vector of error terms. The only assumption on the distribution of the errors is that it is continuous; in that sense the model is general. Recall that the least squares estimator is the minimizor of Euclidean distance between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}_{LS} = \boldsymbol{X}\hat{\boldsymbol{\beta}}_{LS}$. To obtain the R estimator, a different measure of distance is used that is based on the dispersion function of Jaeckel (1972). Jaeckel's dispersion function is given by

$$D(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_\varphi \tag{3}$$

where $\|\cdot\|_\varphi$ is a pseudo-norm defined as

$$\|\boldsymbol{u}\|_\varphi = \sum_{i=1}^{n} a(R(u_i))u_i,$$

where R denotes rank, $a(t) = \varphi\left(\frac{t}{n+1}\right)$, and φ is a non-decreasing, square-integrable *score* function defined on the interval $(0,1)$. Assume without loss of generality that it is standardized, so that $\int \varphi(u)\,du = 0$ and $\int \varphi^2(u)\,du = 1$. Score functions are discussed further in a later section.

The R estimator of $\boldsymbol{\beta}$ is defined as

$$\hat{\boldsymbol{\beta}}_\varphi = \text{Argmin}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_\varphi. \tag{4}$$

This estimator is a highly efficient estimator which is robust in the Y-space. A weighted version can attain 50% breakdown in the X-space at the expense of a loss in efficiency (Chang et al., 1999).

Inference

Under assumptions outlined in the previous section, it can be shown that the solution to (4) is consistent and asymptotically normal (Hettmansperger and McKean, 2011). We summarize this result as follows:

$$\hat{\boldsymbol{\beta}}_\varphi \sim N\left(\boldsymbol{\beta}, \tau_\varphi^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\right)$$

where τ_φ is a scale parameter which depends on f and the score function φ. An estimate of τ_φ is necessary to conduct inference and **Rfit** implements the consistent estimator proposed by Koul et al. (1987). Denote this estimator by $\hat{\tau}_\varphi$. Then Wald tests and confidence regions/intervals can be calculated. Let $\text{se}(\hat{\beta}_j) = \hat{\tau}_\varphi\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}_{jj}$ where $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}_{jj}$ is the jth diagonal element of $\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$. Then an approximate $(1-\alpha)*100\%$ confidence interval for β_j is

$$\hat{\beta}_j \pm t_{1-\alpha/2,n-p-1}\text{se}(\hat{\beta}_j).$$

A Wald test of the general linear hypothesis

$$H_0 : \boldsymbol{M}\boldsymbol{\beta} = \mathbf{0} \text{ versus } H_A : \boldsymbol{M}\boldsymbol{\beta} \neq \mathbf{0}$$

is to reject H_0 if

$$\frac{(\boldsymbol{M}\hat{\boldsymbol{\beta}}_\varphi)^T[\boldsymbol{M}(\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{M}^T]^{-1}(\boldsymbol{M}\hat{\boldsymbol{\beta}})/q}{\hat{\tau}_\varphi^2} > F_{1-\alpha,q,n-p-1}$$

where $q = \dim(\boldsymbol{M})$. Similar to the reduced model test of classical regression rank-based regression offers a *drop in dispersion* test which is implemented in the R function `drop.test`. For the above hypotheses let $\hat{\boldsymbol{\theta}}_\varphi$ be the rank-based coefficient estimate of the reduced model [Model (1) constrained by H_0]. As discussed in Theorem 3.7.2 of Hettmansperger and McKean (2011), the reduced model design matrix is easily obtained using a QR-decomposition on $\boldsymbol{M}^T$. We have implemented this methodology in **Rfit**. Similar to the LS reduction in sums of squares, the rank-based test is based on a reduction of dispersion from the reduced to the full model. Let $D(\hat{\boldsymbol{\theta}}_\varphi)$ and $D(\hat{\boldsymbol{\beta}}_\varphi)$ denote the reduced and full model minimum dispersions, then the test is to reject H_0 if

$$\frac{[D(\hat{\boldsymbol{\theta}}_\varphi) - D(\hat{\boldsymbol{\beta}}_\varphi)]/q}{\hat{\tau}_\varphi/2} > F_{1-\alpha,q,n-p-1}$$

Computation

It can be shown that Jaeckel's dispersion function (3) is a convex function of $\boldsymbol{\beta}$ (Hettmansperger and McKean, 2011). **Rfit** uses `optim` with option `` `BFGS' `` to minimize the dispersion function. We investigated other minimization methods (e.g. Nelder-Mead or CG), however the quasi-Newton method works well in terms of speed and convergence. An orthonormal basis matrix, for the space spanned by the columns

of $\boldsymbol{X}$, is first calculated using `qr` which leads to better performance in examples. The default initial fit is based on an L_1 fit using **quantreg** (Koenker, 2011).

Computations by **Rfit** of rank-based estimation and associated inference are illustrated in the examples of the next section.

Rfit computations of rank-based fitting of linear models

For the general linear model (1) the package **Rfit** obtains the rank-based estimates and inference, as described in the previous section. In this section we illustrate this computation for two examples. The first is for a simple linear model while the second is for a multiple regression model.

Example 1 (Telephone data)**.** We begin with a simple linear regression example using the telephone data discussed in Rousseuw and Leroy (1987) These data represent the number of telephone calls (in tens of millions) placed in Belgium over the years 1950–1973. The data are plotted in Figure 1. There are several noticeable outliers which are due to a mistake in the recording units for the years 1964–1969. This is a simple dataset, containing only one explanatory variable, however it allows us to easily highlight the package and also demonstrate the robustness to outliers of the procedure. The main function of the package **Rfit** is `rfit` which, as the following code segment illustrates, uses syntax similar to `lm`.

```
> library(Rfit)
> data(telephone)
> fit <- rfit(calls ~ year, data = telephone)
> summary(fit)


Call:
rfit(formula = calls ~ year, data = telephone)

Coefficients:
        Estimate  Std. Error t.value p.value
    -284.313842  152.687751 -1.8621 0.07665 .
year    0.145861    0.077842  1.8738 0.07494 .
---
Signif. codes:
     0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared (Robust): 0.3543158
Reduction in Dispersion Test:
     12.07238 p-value: 0.00215


> plot(telephone)
> abline(fit)
> abline(lm(calls ~ year, data = telephone),
+     col = 2, lty = 2)
> legend("topleft", legend = c("R", "LS"),
+     col = 1:2, lty = 1:2)
```

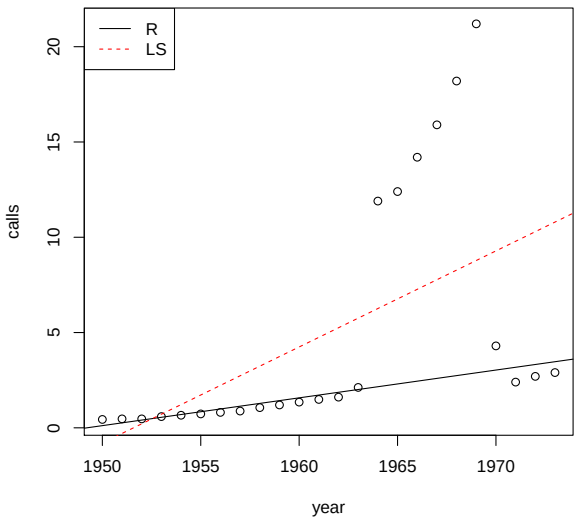


Figure 1: Scatter plot of the telephone data with overlaid regression lines.

Further, the output is similar to that of `lm` and can be interpreted in the same way. The estimate of slope is 0.146 (tens of millions of calls per year) with a standard error of 0.078. The t-statistic is the ratio of the two and the p-value is calculated using a t-distribution with $n-2$ degrees of freedom. Hence one could conclude that year is a marginally significant predictor of the number of telephone calls.

The overlaid fitted regression lines in the scatter plot in Figure 1 demonstrate the robustness of the Wilcoxon fit and the lack of robustness of the least squares fit.

Example 2 (Free fatty acid data)**.** This is a data set from Morrison (1983, p. 64) (c.f. Example 3.9.4 of Hettmansperger and McKean (2011)). The response variable is level of free fatty acid in a sample of prepubescent boys. The explanatory variables are age (in months), weight (in lbs), and skin fold thickness. For this discussion, we chose the Wilcoxon (default) scores for **Rfit**. Based on the residual and Q-Q plots below, however, the underlying error distribution appears to be right-skewed. In a later section we analyze this data set using more appropriate (bent) scores for a right-skewed distribution.

To begin with we demonstrate the reduction in dispersion test discussed in the previous section.

```
> fitF <- rfit(ffa ~ age + weight + skin,
+     data = ffa)
> fitR <- rfit(ffa ~ skin, data = ffa)
> drop.test(fitF, fitR)


Drop in Dispersion Test
F-Statistic     p-value
 1.0754e+01  2.0811e-04
```

As the code segment shows, the syntax is similar to that of the `anova` function used for reduced model testing in many of the parametric packages.

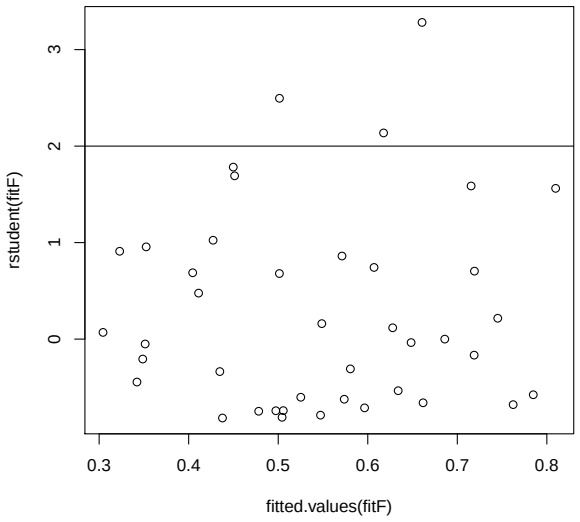


Figure 2: Studentized residuals versus fitted values for the free fatty acid data.

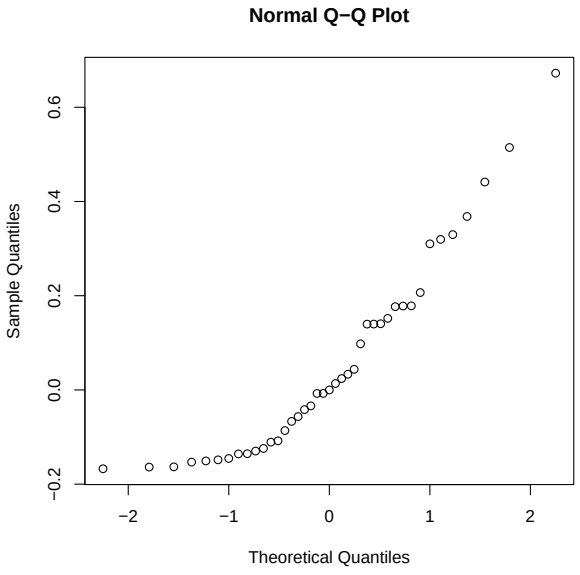


Figure 3: Normal Q-Q plot of the studentized residuals for the free fatty acid data.

Studentized residuals for rank-based fits are calculated in a way similar to the LS studentized residuals (see Chapter 3 of Hettmansperger and McKean, 2011). We have implemented these residuals in **Rfit** and demonstrate their use next. These are the residuals plotted in the residual and Q-Q plots in Figure 2 and Figure 3 respectively. The code is similar to that of least squares analysis. The function `fitted.values` returns the fitted values and `residuals` returns the residuals from the full model fit. The function `rstudent` calculates the studentized residuals.

Common diagnostic tools are the residual plot (Figure 2)

```
> plot(fitted.values(fitF), rstudent(fitF))
> abline(h = c(-2, 2))
```

and normal probability plot of the studentized residuals (Figure 3).

```
> qqnorm(residuals(fitF))
```

As is shown in the plots, there are several outliers and perhaps the errors are from a right skewed distribution. We revist this example in a later section.

One-way ANOVA

Suppose we want to determine the effect that a single factor A has on a response of interest over a specified population. Assume that A consists of k levels or treatments. In a completely randomized design (CRD), n subjects are randomly selected from the reference population and n_i of them are randomly assigned to level i, $i = 1,\ldots k$. Let the jth response in the ith level be denoted by Y_{ij}, $j = 1,\ldots$, $i = 1,\ldots,k$. We assume that the responses are independent of one another and that the distributions among levels differ by at most shifts in location.

Under these assumptions, the full model can be written as

$$Y_{ij} = \mu_i + e_{ij} \quad j = 1,\ldots,n_i \;\; i = 1,\ldots,k\,, \tag{5}$$

where the e_{ij}s are iid random variables with density $f(x)$ and the parameter μ_i is a convenient location parameter for the ith level, (for example, the mean or median of the ith level). Generally, the parameters of interest are the effects (contrasts), $\Delta_{ii'} = \mu_{i'} - \mu_i$, $i \neq i', 1,\ldots,k$. Upon fitting the model a residual analysis should be conducted to check these model assumptions.

Observational studies can also be modeled this way. Suppose k independent samples are drawn from k different populations. If we assume further that the distributions for the different populations differ by at most a shift in locations then Model (5) is appropriate.

The analysis for this design is usually a test of the hypothesis that all the effects are 0, followed by individual comparisons of levels. The hypothesis can be written as

$$\begin{aligned} H_0 &: \mu_1 = \cdots = \mu_k \quad \text{versus} \\ H_A &: \mu_i \neq \mu_{i'} \quad\;\; \text{for some } i \neq i'. \end{aligned} \tag{6}$$

Confidence intervals for the simple contrasts $\Delta_{ii'}$ are generally used to handle the comparisons. **Rfit** offers

a reduction in dispersion test for testing (6) as well as pairwise p-values adjusted for multiple testing. The function `oneway.rfit` is illustrated in the following example.

Example 3 (LDL cholesterol of quail)**.** Hettmansperger and McKean (2011, p. 295) discuss a study that investigated the effect of four drug compounds on low density lipid (LDL) cholesterol in quail. The drug compounds are labeled as I, II, III, and IV. The sample size for each of the first three levels is 10 while 9 quail received compound IV. The boxplots shown in Figure 4 attest to a difference in the LDL levels.

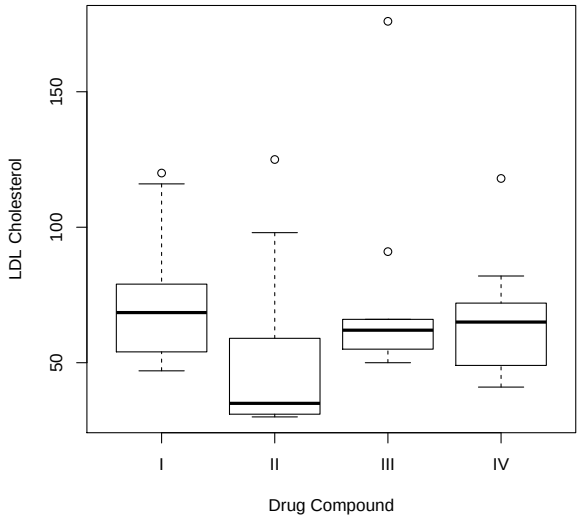


Figure 4: Comparison boxplots for quail data.

Using Wilcoxon scores, we fit the full model. The summary of the test of hypotheses (6) as computed by the **Rfit** function `oneway.rfit` follows. The resulting Q-Q plot (Figure 5) of the studentized residuals indicates that the random errors e_{ij} have a skewed distribution.

```
> data(quail)
> oneway.rfit(quail$ldl, quail$treat)

Call:
oneway.rfit(y = quail$ldl, g = quail$treat)

Overall Test of All Locations Equal

Drop in Dispersion Test
F-Statistic     p-value
   3.916404    0.016403

        Pairwise comparisons using Rfit

data:  quail$ldl and quail$treat

  2    3    4
1 -    -    -
2 1.00 -    -
3 0.68 0.99 -
4 0.72 0.99 0.55

P value adjustment method: none
```

Robust fits based on scores more appropriate than the Wilcoxon for skewed errors are discussed later. Note that the results from a call to `oneway.rfit` include the results from the call to `rfit`.

```
> anovafit <- oneway.rfit(quail$ldl, quail$treat)
```

Which may then be used for diagnostic procedures, such as the Q-Q plot of the studentized residuals in Figure 5.

```
> qqnorm(rstudent(anovafit$fit))
```

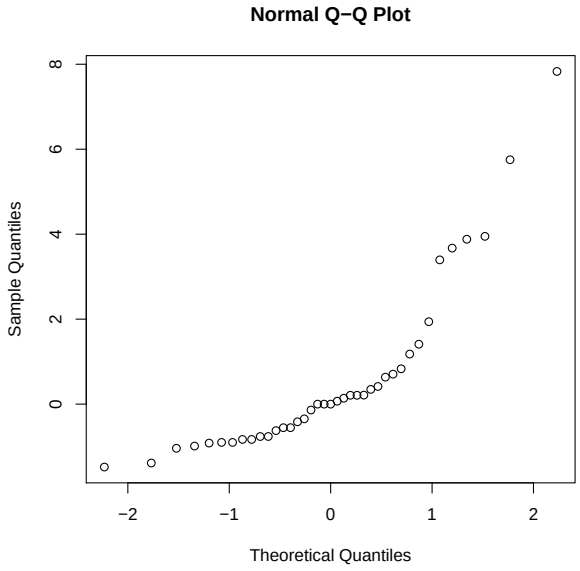


Figure 5: Normal Q-Q plot of the studentized residuals for the quail data.

With a p-value of 0.0164, generally, the null hypothesis would be rejected and the inference would pass to the comparisons of interest. Finally, we note that, the LS test of the null hypothesis has p-value 0.35; hence, with the LS analysis H_0 would not be rejected. In practice, one would not proceed with comparisons with such a large p-value. Thus, for this data set the robust and LS analyses have different interpretations.

Multiple comparisons

The second stage of an analysis of a one-way design usually consists of pairwise comparisons of the treatments. The robust $(1-\alpha)100\%$ confidence interval to compare the ith and $i'th$ treatments is given by

$$\widehat{\Delta}_{ii'} \pm t_{\alpha/2,n-1}\widehat{\tau}_{\varphi}\sqrt{\frac{1}{n_i}+\frac{1}{n_{i'}}}. \qquad (7)$$

Often there are many comparisons of interest. For example, in the case of all pairwise comparisons there

are $\binom{k}{2}$ confidence intervals. Hence, the overall family error rate is usually of concern. Multiple comparison procedures (MCP) try to control the overall error rate to some degree. There are many MCPs from which to choose; see Chapter 4 of Hettmansperger and McKean (2011) for a review of many of these procedures from a robust perspective. In **Rfit** we supply a summary function that adjusts confidence intervals and use three of the most popular such procedures: protected least significant difference (none); Tukey-Kramer (tukey); and the Bonferroni (bonferroni). These methods are described in many standard statistics texts.

Example 4 (LDL cholesterol of quail, continued)**.** For the quail data, we selected the Tukey-Kramer procedure for all six pairwise comparisons. Use of the code and example output is given below. The multiple comparison part of the output is:

```
> summary(oneway.rfit(quail$ldl, quail$treat),
+     method = "tukey")

Multiple Comparisons
Method Used  tukey

  I J  Estimate  St Err  Lower CI Upper CI
1 1 2 -25.00720 8.26813 -47.30553 -2.70886
2 1 3  -3.99983 8.26813 -26.29816 18.29851
3 1 4  -5.00027 8.49469 -27.90963 17.90909
4 2 3 -21.00737 8.26813 -43.30571  1.29096
5 2 4 -20.00693 8.49469 -42.91629  2.90243
6 3 4   1.00044 8.49469 -21.90892 23.90981
```

The Tukey-Kramer procedure declares that the Drug Compounds I and II differ significantly.

Multi-way ANOVA

In this section, we consider a k-way crossed factorial experimental design. For these designs, the **Rfit** function `raov` computes the rank-based analysis for all $2^k - 1$ hypotheses including the main effects and interactions of all orders. The design may be balanced or unbalanced. For simplicity, we briefly discuss the analysis in terms of a cell mean (median) model; see Hocking (1985) for details on the traditional LS analysis and Chapter 4 of Hettmansperger and McKean (2011) for the rank-based analysis. For this paper, we illustrate **Rfit** using a two-way crossed factorial design, but similarly **Rfit** computes the rank-based analysis of a k-way design.

Let A and B denote the two factors with levels a and b, respectively. Let Y_{ijk} denote the response for the kth replication at levels i and j of factors A and B, respectively. Then the full model can be expressed as

$$Y_{ijk} = \mu_{ij} + e_{ijk} \quad \begin{array}{l} k = 1 \ldots n_{ij} \\ i = 1 \ldots a \\ j = 1 \ldots b, \end{array} \tag{8}$$

where e_{ijk} are iid random variables with pdf $f(t)$. Since the effects of interest are contrasts in the μ_{ij}'s, these parameters can be either cell means or medians, (actually any location functional suffices). **Rfit** implements a reduction in dispersion tests for testing all main effects and interactions.

For the two-way model, the three hypotheses of immediate interest are the main effects hypotheses and the interaction hypothesis. We have chosen Type III hypotheses which are easy to interpret even for severely unbalanced designs. Following Hocking (1985), the hypothesis matrices $\boldsymbol{M}$ can easily be computed in terms of Kronecker products. As discussed in a previous section, for these tests the drop in dispersion test statistics can easily be constructed. We have implemented this formulation in **Rfit**.

Example 5 (Box-Cox data)**.** Consider the data set discussed by Box and Cox (1964). The data are the results of a 3×4 two-way design, where forty-eight animals were exposed to three different poisons and four different treatments. The design is balanced with four replications per cell. The response was the log survival time of the animal. An interaction plot using the cell medians is presented in Figure 6. Obviously the profiles are not parallel and interaction is present.

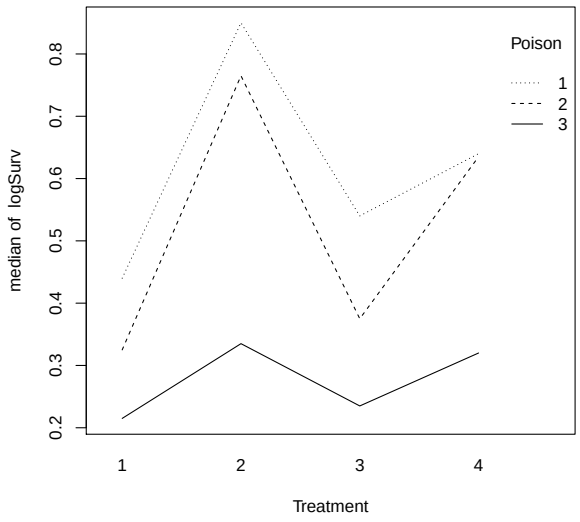


Figure 6: Interaction Plot for Box-Cox Data.

The output below displays the Wilcoxon ANOVA table, which indicates that interaction is highly significant, $p = 0.0143$, confirming the profile plot. On the other hand, the LS test F statistic for interaction is 1.87 with $p = 0.1123$. Hence, the LS test fails to detect interaction.

```
> data(BoxCox)
> attach(BoxCox)
> fit <- raov(logSurv ~ Treatment + Poison)
> fit
```

```
Robust ANOVA Table
    DF        RD   Mean RD         F      p-value
T    3 2.9814770 0.9938257 21.263421 4.246022e-08
P    2 3.6987828 1.8493914 39.568699 8.157360e-10
T:P  6 0.8773742 0.1462290  3.128647 1.428425e-02
```

Writing score functions for Rfit

As discussed earlier, we must choose a score function for rank-based fitting. For most datasets the default option of Wilcoxon scores works quite well, however, occasionally choosing a different score function can lead to a more efficient analysis. In this section we first discuss score functions in general and then illustrate how the user may create his own score function. We have placed the score functions in an object of class `"scores"`. A `"scores"` object consists of two objects of type function and an optional numeric object. The functions are the score function `phi` and it's derivative `Dphi`. The derivative is necessary in estimating τ_φ. Below is what the class for Wilcoxon scores looks like.

```
> wscores

An object of class "scores"
Slot "phi":
function(u) sqrt(12)*(u-0.5)

Slot "Dphi":
function(u) rep(sqrt(12),length(u))

Slot "param":
NULL
```

Other score functions included in **Rfit** are listed in Table 1. A plot of the bent score functions is provided in Figure 7. Other score functions can be plotted by getting the scores using the method `getScores`. For example the commands `u<-seq(0.01,0.99,by=0.01)` `plot(u,getScores(nscores,u))` graphs the normal scores.

Score	Keyword	Recommended usage
Wilcoxon	`wscores`	moderate tailed
Normal	`nscores`	light-moderate tailed
Bent1	`bentscores1`	highly right skewed
Bent2	`bentscores2`	light tailed
Bent3	`bentscores3`	highly left skewed
Bent4	`bentscores4`	moderately heavy tailed

Table 1: Table of available score functions. Unless otherwise noted, distribution is assumed to be symmetric.

Next we illustrate how to create the score function for the *bent* scores. Bent scores are recommended when the errors come from a skewed distribution. An appropriate bent score function for skewed distribution with a right heavy tail is

$$\phi(u) = \begin{cases} 4u - 1.5 & \text{if } u \leq 0.5 \\ 0.5 & \text{if } u > 0.5 \end{cases}$$

The following code segment defines the scores.

```
> bent.phi <- function(u, ...)
+     ifelse(u < 0.5, 8/3 * u - 1, 1/3)
> bent.Dphi <- function(u, ...)
+     ifelse(u < 0.5, 8/3, 0)
> bentscores <- new("scores", phi = bent.phi,
+     Dphi = bent.Dphi)
```

They are displayed graphically in the top left quadrant of Figure 7.

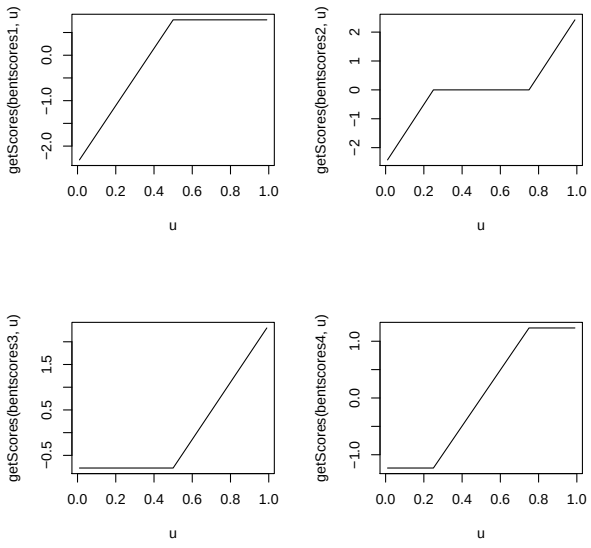


Figure 7: Plots of four bent score functions.

Below we implement the newly defined score functions using the free fatty acid data previously analysed using Wilcoxon scores. One could also use the scores provided by **Rfit** with option `scores=bentscores1` to obtain the same result.

```
> summary(rfit(ffa ~ age + weight + skin,
+     scores = bentscores, data = ffa))

Call:
rfit.default(formula = ffa ~ age + weight + skin,
    scores = bentscores, data = ffa)

Coefficients:
          Estimate  Std. Error t.value   p.value
        1.35957548  0.18882744  7.2001 1.797e-08 ***
age    -0.00048157  0.00178449 -0.2699 0.7888044
weight -0.01539487  0.00260504 -5.9097 9.176e-07 ***
skin    0.35619596  0.09090132  3.9185 0.0003822 ***
---
Signif. codes:
     0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Multiple R-squared (Robust): 0.4757599
Reduction in Dispersion Test:
     11.19278 p-value: 2e-05
```

The results are similar to those presented in Hettmansperger and McKean (2011).

Summary and future work

This paper illustrates the usage of a new R package, **Rfit**, for rank-based estimation and inference. Rank-based methods are robust to outliers and offer the data analyst an alternative to least squares. **Rfit** includes algorithms for general scores and a library of score functions is included. Functions for regression as well as one-way and multi-way anova are included. We illustrated the use of **Rfit** on several real data sets.

We are in the process of extending **Rfit** to include other robust rank-based procedures which are discussed in Chapters 3 and 5 of Hettmansperger and McKean (2011). These include autoregressive time-series models, cluster correlated data (mixed models), and nonlinear models. We are also developing weighted versions of rank-based estimation that can be used in mixed effects modeling as discussed in Kloke et al. (2009) as well as the computation of high breakdown rank-based estimation discussed in Chang et al. (1999).

Bibliography

G. Box and D. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964. [p62]

W. H. Chang, J. W. McKean, J. D. Naranjo, and S. J. Sheather. High-breakdown rank regression. *Journal of the American Statistical Association*, 94(445): 205–219, 1999. ISSN 0162-1459. [p57, 58, 64]

K. S. Crimin, A. Abebe, and J. W. McKean. Robust general linear models and graphics via a user interface. *Journal of Modern Applied Statistics*, 7:318–330, 2008. [p57]

T. P. Hettmansperger and J. W. McKean. *Robust Nonparametric Statistical Methods, 2nd Ed.* Chapman Hall, New York, 2011. ISBN 0-340-54937-8; 0-471-19479-4. [p57, 58, 59, 60, 61, 62, 64]

R. R. Hocking. *The Analysis of Linear Models*. Brooks/Cole, Monterey, CA, 1985. [p62]

M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods, Second Edition*. John Wiley & Sons, New York, New York, 1999. [p57]

L. A. Jaeckel. Estimating regression coefficients by minimizing the dispersion of the residuals. *The Annals of Mathematical Statistics*, 43:1449–1458, 1972. [p57, 58]

J. Jurečková. Nonparametric estimate of regression coefficients. *The Annals of Mathematical Statistics*, 42:1328–1338, 1971. [p57]

J. Kapenga, J. W. McKean, and T. J. Vidmar. RGLM: Users manual. Technical Report 90, Western Michigan University, Department of Mathematics and Statistics, 1995. [p57]

J. Kloke, J. McKean, and M. Rashid. Rank-based estimation and associated inferences for linear models with cluster correlated errors. *Journal of the American Statistical Association*, 104(485):384–390, 2009. [p57, 64]

R. Koenker. *quantreg: Quantile Regression*, 2011. URL http://CRAN.R-project.org/package=quantreg. R package version 4.69. [p59]

H. L. Koul, G. Sievers, and J. W. McKean. An estimator of the scale parameter for the rank analysis of linear models under general score functions. *Scandinavian Journal of Statistics*, 14:131–141, 1987. [p58]

J. McKean. Robust analysis of linear models. *Statistical Science*, 19(4):562–570, 2004. [p57]

J. McKean and T. Hettmansperger. A robust analysis of the general linear model based on one step R-estimates. *Biometrika*, 65(3):571, 1978. [p57]

J. McKean and S. Sheather. Diagnostic procedures. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):221–233, 2009. [p57]

J. McKean, J. Terpstra, and J. Kloke. Computational rank-based statistics. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):132–140, 2009. [p57]

D. F. Morrison. *Applied Linear Statistical Models*. Prentice Hall, Englewood Cliffs, 1983. [p59]

P. J. Rousseuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987. [p59]

J. Terpstra and J. W. McKean. Rank-based analyses of linear models using R". *Journal of Statistical Software*, 14(7), 2005. [p57]

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. [p57]

John D. Kloke
Department of Biostatistics and Medical Informatics
University of Wisconsin-Madison
Madison, WI 53726
kloke@biostat.wisc.edu

Joseph W. McKean
Department of Statistics
Western Michigan University
Kalamazoo, MI 49008
joseph.mckean@wmich.edu

Graphical Markov Models with Mixed Graphs in R

by Kayvan Sadeghi and Giovanni M. Marchetti

Abstract In this paper we provide a short tutorial illustrating the new functions in the package **ggm** that deal with ancestral, summary and ribbonless graphs. These are mixed graphs (containing three types of edges) that are important because they capture the modified independence structure after marginalisation over, and conditioning on, nodes of directed acyclic graphs. We provide functions to verify whether a mixed graph implies that A is independent of B given C for any disjoint sets of nodes and to generate maximal graphs inducing the same independence structure of non-maximal graphs. Finally, we provide functions to decide on the Markov equivalence of two graphs with the same node set but different types of edges.

Introduction and background

Graphical Markov models have become a part of the mainstream of statistical theory and application in recent years. These models use graphs to represent conditional independencies among sets of random variables. Nodes of the graph correspond to random variables and edges to some type of conditional dependency.

Directed acyclic graphs

In the literature on graphical models the two most used classes of graphs are *directed acyclic graphs* (DAGs) and *undirected graphs*. DAGs have proven useful, among other things, to specify the data generating processes when the variables satisfy an underlying partial ordering.

For instance, suppose that we have four observed variables: Y, the ratio of systolic to diastolic blood pressure and X the diastolic blood pressure, both on a log scale; Z, the body mass and W, the age, and that a possible generating process is the following linear recursive regression model

$$\begin{aligned} Y &= \gamma_{YZ}Z + \gamma_{YU}U + \epsilon_Y \\ X &= \gamma_{XW}W + \gamma_{XU}U + \epsilon_X \\ Z &= \gamma_{ZV}W + \epsilon_Z \\ W &= \epsilon_W;\ U = \epsilon_U, \end{aligned}$$

where all the variables are mean-centered and the ϵs are zero mean, mutually independent Gaussian random errors. In this model we assume that there exists a genetic factor U influencing the ratio and levels of blood pressure.

This model can be represented by the DAG in Figure 1(a) with nodes associated with the variables and edges indicating the dependencies represented by the regression coefficients (γs).

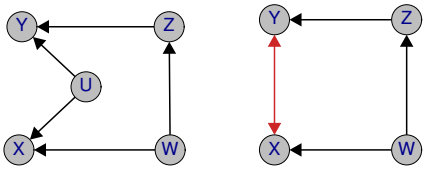


Figure 1: (a) A DAG. (b) A regression chain graph.

From the graph it is seen, for instance, that the ratio of the two blood pressures (Y) is directly influenced by body mass (Z) but not by age (W). Thus a consequence of the model is that the variables must satisfy a set of conditional independencies: for example, the ratio of the blood pressure is independent of the age given the body mass, written as $Y \perp\!\!\!\perp W|Z$.

A remarkable result is that the independencies can be deduced from the graph alone, without reference to the equations, by using a criterion called d-separation. In fact, in the graph of Figure 1(a), the nodes Y and W are d-separated given Z. This can be checked using special graph algorithms included, for example, in packages **gRain** (Højsgaard, 2012) and **ggm** (Marchetti et al., 2012). For more details on DAG models and their implementation in R see the extensive discussion in Højsgaard et al. (2012).

Hidden variables and induced graphs

The model has four observed variables but includes an unobserved variable, that is, the genetic factor U. When U is hidden the model for the observed variables becomes

$$\begin{aligned} Y &= \gamma_{YZ}Z + \eta_Y \\ X &= \gamma_{XW}W + \eta_X \\ Z &= \gamma_{ZV}W + \epsilon_Z \\ W &= \epsilon_W; \end{aligned}$$

with two correlated errors $\eta_Y = \gamma_{YU}U + \epsilon_Y$ and $\eta_X = \gamma_{XU}U + \epsilon_X$, such that $\mathrm{cov}(\eta_Y, \eta_X) = \omega_{YX}$. As a consequence the model is still a recursive model and the parameters have a regression parameter interpretation, but contain some correlated residuals.

The induced model is said to be obtained after marginalisation over U. In this model some of the original independencies are lost, but we can observe the implied independencies $Y \perp\!\!\!\perp W|Z$ and $X \perp\!\!\!\perp Z|W$. Also it can be shown that it is impossible to represent such independencies in a DAG model defined for the

four observed variables. Therefore, we say that DAG models are not *stable* under marginalisation.

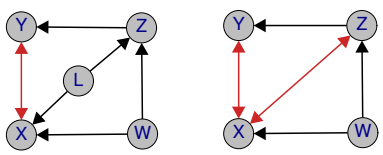
A mixed graph with arrows and arcs, as shown in Figure 1(b), can be used to represent the induced independence model after marginalisation over U. In this representation, beside the arrows, represented by the γs, we have the arc $Y \prec\!\!\succ X$ associated with the (partial) correlation ω_{YX}.

The graph of Figure 1(b) belongs to a class of models called *regression chain graph models*. This class generalises the recursive generating process of DAGs by permitting joint responses, coupled in the graph by arcs, and thus appears to be an essential extension for applications; see Cox and Wermuth (1996). Regression chain graphs can be used as a conceptual framework for understanding multivariate dependencies, for example in longitudinal studies. The variables are arranged in a sequence of blocks, such that (a) all variables in one block are of equal standing and any dependence between them is represented by an arc, and (b) all variables in one block are responses to variables in all blocks to their right, so that any dependencies between them are directed, represented by an arrow pointing from right to left. The graph shows how the data analysis can be broken down into a series of regressions and informs about which variables should or should not be controlled for in each regression.

More general induced graphs

The class of regression chain graphs is not, however, stable under marginalisation. For instance, suppose that the generating process for the blood pressure data is defined by the more general regression chain graph of Figure 2(a) where L is a further variable representing a common hidden cause of systolic blood pressure and body mass.

Then, after marginalisation over L, the model can still be described by a linear system of equations with correlated residuals and can be represented by the mixed graph shown in Figure 2(b). But the resulting graph is not a DAG nor a regression chain graph because it contains the pair of variables (Y, Z) coupled by both a directed edge and a path composed by bi-directed arcs. Thus Y cannot be interpreted as a pure response to Z and in addition Y and Z are not two joint responses.



(a) (b)

Figure 2: (a) A regression chain graph model; (b) the mixed graph obtained after marginalisation over L, which is not a regression chain graph.

Stable mixed graphs

The previous illustrations show that when there are unobserved variables, DAG or regression chain graph models are no longer appropriate. The discussion could be extended to situations where there are some selection variables that are hidden variables that are conditioned on.

This motivates the introduction of a more general class of *mixed graphs*, which contains three types of edges, denoted by lines, $\multimap$, arrows, $\multimap\!\!\succ$, and arcs (bi-directed arrows), $\prec\!\!\succ$. In the case of regression models, explained above, lines generally link pairs of joint context (explanatory) variables and arcs generally link pairs of joint response variables.

There are at least three known classes of mixed graphs without self loops that remain in the same class, i.e. that are *stable under marginalisation and conditioning*. The largest one is that of *ribbonless graphs* (RGs) (Sadeghi, 2012a), defined as a modification of MC-graphs (Koster, 2002). Then, there is the subclass of *summary graphs* (SGs) (Wermuth, 2011), and finally the smallest class of the *ancestral graphs* (AGs) (Richardson and Spirtes, 2002).

Four tasks of the current paper

In this paper, we focus on the implementation of four important tasks performed on the class of mixed graphs in R:

1. Generating different types of stable mixed graphs after marginalisation and conditioning.
2. Verifying whether an independency of the form $Y \perp\!\!\!\perp W | Z$ holds by using a separation criterion called m-separation.
3. Generating a graph that induces the same independence structure as an input mixed graph such that the generated graph is *maximal*, i.e. each missing edge of the generated graph implies at least an independence statement.
4. Verifying whether two graphs are *Markov equivalent*, i.e. they induce the same independencies, and whether, given a graph of a specific type, there is a graph of a different type that is Markov equivalent to it.

Package ggm

The tasks above are illustrated by using a set of new functions introduced into the R package **ggm** (Marchetti et al., 2012). In the next section we give the details of how general mixed graphs are defined. The following four sections deal with the four tasks respectively. For each task we give a brief introduction at the beginning of its corresponding section.

Some of the functions generalise previous contributions of **ggm** discussed in Marchetti (2006). The

ggm package has been improved and it is now more integrated with other contributed packages related to graph theory, such as **graph** (Gentleman et al., 2012), **igraph** (Csardi and Nepusz, 2006), and **gRbase** (Dethlefsen and Højsgaard, 2005), which are now required for representing and plotting graphs. Specifically, in addition to adjacency matrices, all the functions in the package now accept `graphNEL` and `igraph` objects as input, as well as a new character string representation. A more detailed list of available packages for graphical models can be found at the CRAN Task View *gRaphical Models in R* at `http://cran.r-project.org/web/views/gR.html`.

Defining mixed graphs in R

For a comprehensive discussion on the ways of defining a directed acyclic graph, see Højsgaard et al. (2012). A mixed graph is a more general graph type with at most three types of edge: directed, undirected and bi-directed, with possibly multiple edges of different types connecting two nodes. In **ggm** we provide some special tools for mixed graphs that are not present in other packages. Here we briefly illustrate some methods to define mixed graphs and we plot them with a new function, `plotGraph`, which uses a Tk GUI for basic interactive graph manipulation.

The first method is based on a generalisation of the adjacency matrix. The second uses a descriptive vector and is easy to use for small graphs. The third uses a special function `makeMG` that allows the directed, undirected, and bi-directed components of a mixed graph to be combined.

Adjacency matrices for mixed graphs

In the adjacency matrix of a mixed graph we code the three different edges with a binary indicator: 1 for directed, 10 for undirected and 100 for bi-directed edges. When there are multiple edges the codes are added.

Thus the *adjacency matrix of a mixed graph* H with node set N and edge set F is an $|N| \times |N|$ matrix obtained as $A = B + S + W$ by adding three matrices $B = (b_{ij})$, $S = (s_{ij})$ and $W = (w_{ij})$ defined by

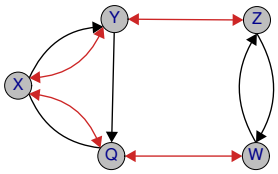
$$b_{ij} = \begin{cases} 1, & \text{if and only if } i \longrightarrow j \text{ in } H; \\ 0, & \text{otherwise.} \end{cases}$$

$$s_{ij} = s_{ji} = \begin{cases} 10, & \text{if and only if } i \text{ —— } j \text{ in } H; \\ 0, & \text{otherwise.} \end{cases}$$

$$w_{ij} = w_{ji} = \begin{cases} 100, & \text{if and only if } i \prec\!\!\longrightarrow\!\!\succ j \text{ in } H; \\ 0, & \text{otherwise.} \end{cases}$$

Notice that because of the symmetric nature of lines and arcs S and W are symmetric, whereas B is not necessarily symmetric.

For instance consider the following general mixed graph.:



Notice that this graph is not of much interest per se, because it is not a stable graph, but it is introduced just to illustrate the structure of the adjacency matrix.

This graph can be defined by the commands

```
> mg <- matrix(c(  0, 101,   0,   0, 110,
                 100,   0, 100,   0,   1,
                   0, 110,   0,   1,   0,
                   0,   0,   1,   0, 100,
                 110,   0,   0, 100,   0),
               5, 5, byrow = TRUE)
> N <- c("X","Y","Z","W","Q")
> dimnames(mg) <- list(N, N)
> mg

    X   Y   Z   W   Q
X   0 101   0   0 110
Y 100   0 100   0   1
Z   0 110   0   1   0
W   0   0   1   0 100
Q 110   0   0 100   0
```

and plotted with `plotGraph(mg)`.

Defining mixed graphs by using vectors

A more convenient way of defining small mixed graphs is based on a simple vector coding as follows. The graph is defined by a character vector of length $3f$, where $f = |F|$ is the number of edges, and the vector contains a sequence of triples $\langle$`type`, `label1`, `label2`$\rangle$, where the `type` is the edge type and `label1` and `label2` are the labels of the two nodes. The edge type accepts `"a"` for a directed arrow , `"b"` for an arc and `"l"` for a line. Notice that isolated nodes may not be created by this method. For example, the vector representation of the previous mixed graph is

```
> mgv <- c("b","X","Y","a","X","Y","l","X","Q",
           "b","Q","X","a","Y","Q","b","Y","Z",
           "a","Z","W","a","W","Z","b","W","Q")
```

Once again as in the DAG case we can use `plotGraph(mgv)` to plot the defined graph.

Mixed graph using the function `makeMG`

Finally the adjacency matrix of a mixed graph may be built up with the function `makeMG`. This function requires three arguments `dg`, `ug` and `bg`, corresponding respectively to the three adjacency matrices B, S and W composing the mixed graph. These may also

be obtained by the constructor functions `DG` and `UG` of **ggm** for directed and undirected graphs respectively. Thus for the previous mixed graph we can issue the command

```
> mg <- makeMG(dg = DG(Y ~ X, Z ~ W, W ~ Z),
               ug = UG(~ X*Q),
               bg = UG(~ Y*X + X*Q + Q*W + Y*Z))
```

obtaining the same adjacency matrix (up to a permutation).

Generating stable mixed graphs

There are four general classes of stable mixed graphs.

The more general class is that of ribbonless graphs: these are mixed graphs without a specific set of subgraphs called ribbons. Figure 3 below shows two examples of ribbons. The exact definition of ribbons is given in Sadeghi (2012a).

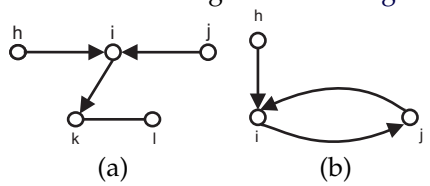


Figure 3: Two commonly seen ribbons $\langle h, i, j \rangle$.

The lack of ribbons ensures that, for any RG, there is a DAG whose independence structure, i.e. the set of all conditional independence statements that it induces after marginalisation over, and conditioning on, two disjoint subsets of its node set can be represented by the given RG. This is essential, as it shows that the independence structures corresponding to RGs are probabilistic, that is, there exists a probability distribution P that is faithful with respect to any RG, i.e. for random vectors X_A, X_B, and X_C with probability distribution P, $X_A \perp\!\!\!\perp X_B \,|\, X_C$ if and only if $\langle A, B \,|\, C \rangle$ is in the induced independence structure by the graph. This probability distribution is the marginal and conditional of a probability distribution that is faithful to the generating DAG.

The other classes of stable graphs are further simplification of the class of ribbonless graphs. Summary graphs have the additional property that there are neither arrowheads pointing to lines (i.e. $\prec\!\!\succ \circ$ —— or $\longrightarrow \circ$ ——) nor directed cycles with all arrows pointing towards one direction.

Ancestral graphs have the same constraints as summary graphs plus the additional prohibition of *bows*, i.e. arcs with one endpoint that is an ancestor of the other endpoint; see Richardson and Spirtes (2002).

However, for some ribbonless and summary graphs the corresponding parametrisation is sometimes not available even in the case of a standard joint Gaussian distribution.

If we suppose that stable mixed graphs are only used to represent the independence structure after marginalisation and conditioning, we can consider all types as equally appropriate. However, each of the three types has been used in different contexts and for different purposes. RGs have been introduced in order to straightforwardly deal with the problem of finding a class of graphs that is closed under marginalisation and conditioning by a simple process of deriving them from DAGs. SGs are used when the generating DAG is known, to trace the effects in the sets of regressions as described earlier. AGs are simple graphs, meaning that they do not contain multiple edges and the lack of bows ensures that they satisfy many desirable statistical properties.

In addition, when one traces the effects in regression models with latent and selection variables (as described in the introduction) ribbonless graphs are more alerting to possible distortions (due to indirect effects) than summary graphs, and summary graphs are more alerting than ancestral graphs; see also Wermuth and Cox (2008). For the exact definition and a thorough discussion of all such graphs, see Sadeghi (2012a).

Sadeghi (2012a) also defines the algorithms for generating stable mixed graphs of a specific type for a given DAG or for a stable mixed graph of the same type after marginalisation and conditioning such that they induce the marginal and conditional DAG-independence structure. We implement these algorithms in this paper.

By "generating graphs" we mean applying the defined algorithms, e.g. those for generating stable mixed graphs to graphs, in order to generate new graphs.

Functions to generate the three main types of stable mixed graphs

Three main functions `RG`, `SG`, and `AG` are available to generate and plot ribbonless, summary, and ancestral graphs from DAGs, using the algorithms in Sadeghi (2012a). These algorithms look for the paths with three nodes and two edges in the graph whose inner nodes are being marginalised over or conditioned on, and generate appropriate edges between the endpoints. These have two important properties: (a) they are well-defined in the sense that the process can be performed in any order and will always produce the same final graph, and (b) the generated graphs induce the modified independence structure after marginalisation and conditioning; see Sadeghi (2012a) for more details.

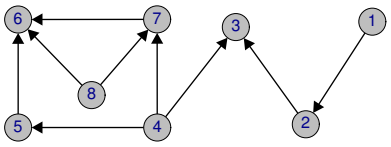
The functions `RG`, `SG`, and `AG` all have three arguments: `a`, the given input graph, `M`, the marginalisation set and `C`, the conditioning set. The graph may be of class `"graphNEL"` or of class `"igraph"` or may be represented by a character vector, or by an adjacency matrix, as explained in the previous sections.

The sets `M` and `C` (default `c()`) must be disjoint vectors of node labels, and they may possibly be empty sets. The output is always the adjacency matrix of the generated graph. There are two additional logical arguments `showmat` and `plot` to specify whether the adjacency matrix must be explicitly printed (default `TRUE`) and the graph must be plotted (default `FALSE`).

Some examples

We start from a DAG defined in two ways, as an adjacency matrix and as a character vector:

```
> ex <- matrix(c(0,1,0,0,0,0,0,0,
                 0,0,1,0,0,0,0,0,
                 0,0,0,0,0,0,0,0,
                 0,0,1,0,1,0,1,0,
                 0,0,0,0,0,1,0,0,
                 0,0,0,0,0,0,0,0,
                 0,0,0,0,0,1,0,0,
                 0,0,0,0,0,1,1,0),
               8, 8, byrow = TRUE)
>
> exvec <- c("a",1,2,"a",2,3,"a",4,3,
             "a",4,5,"a",4,7,"a",5,6,
             "a",7,6,"a",8,6,"a",8,7)


> plotGraph(ex)
```

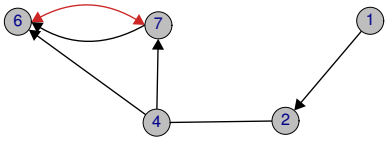


Then we define two disjoint sets M and C to marginalise over and condition on

```
> M <- c(5,8)
> C <- 3
```

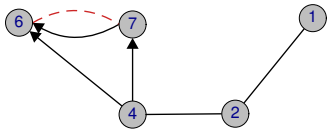
and we generate the ribbonless, summary and ancestral graphs from the DAG with the associated plot.

```
> RG(ex, M, C, plot = TRUE)

  1  2  4   6   7
1 0  1  0   0   0
2 0  0 10   0   0
4 0 10  0   1   1
6 0  0  0   0 100
7 0  0  0 101   0
```



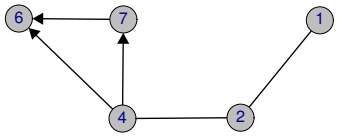
The summary graph is also plotted:

```
> plotGraph(SG(ex,M,C))

   1  2  4   6   7
1  0 10  0   0   0
2 10  0 10   0   0
4  0 10  0   1   1
6  0  0  0   0 100
7  0  0  0 101   0
```



The induced ancestral graph is obtained from the DAG defined as a vector.

```
> AG(exvec, M, C, showmat = FALSE, plot = TRUE)
```



Verifying m-separation

To globally verify whether an independence statement of the form $A \perp\!\!\!\perp B \mid C$ is implied by a mixed graph we use a separation criterion called *m-separation*. This has been defined in Sadeghi (2012a) for the general class of loopless mixed graphs and is the same as the m-separation criterion defined in Richardson and Spirtes (2002) for ancestral graphs. It is also a generalisation of the d-separation criterion for DAGs (Pearl, 1988). This is a graphical criterion that looks to see if the graph contains special paths connecting two sets A and B and involving a third set C of the nodes. These special paths are said to be active or m-connecting. For example, a directed path from a node in A to a node in B that does not contain any node of C is m-connecting A and B. However, if such a path intercepts a node in C then A and B are said to be m-separated given C. However, this behaviour can change if the path connecting A and B contains a collision node or a *collider* for short, that is a node c where the edges meet head-to-head, e.g. $\longrightarrow c \prec\!\!\!-\!\!\!-$ or $\longrightarrow c \prec\!\!\!-\!\!\!-\!\!\!\succ$.

In general, a path is said to be *m-connecting* given C if all its collider nodes are in C or in the set of ancestors of C, and all its non-collider nodes are outside C. For two disjoint subsets A and B of the node set, we say that C m-separates A and B if there is no m-connecting path between A and B given C.

Function for verifying m-separation

The m-separation criterion has been implemented in **ggm** and is available by using the function `msep`.

Note that there is still a function `dSep` in **ggm** for *d*-separation, although it is superseded by `msep`.

The function has four arguments, where the first is the graph `a`, in one of the forms discussed before, and the other three are the disjoint sets `A`, `B`, and `C`.

Examples

For example, consider the DAG of Figure 1(a):

```
> a <- DAG(Y ~ U + Z, X ~ U + W, Z ~ W)
```

We see that Y and W are m-separated given Z:

```
> msep(a, "Y", "W", "Z")

[1] TRUE
```

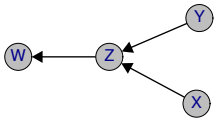
and the same statement holds for the induced ancestral graph after marginalisation over `U`:

```
> b <- AG(a, M = "U")
> msep(b, "Y", "W", "Z")

[1] TRUE
```

This was expected because the induced ancestral graph respects all the independence statements induced by m-separation in the DAG, and not involving the variable `U`.

As a more complex example, consider the following summary graph,

```
> a <- makeMG(dg= DG(W ~ Z, Z ~ Y + X),
              bg= UG(~ Y*Z))

> plotGraph(a)
```



Then, the two following statements verify whether `X` is m-separated from `Y` given `Z`, and whether `X` is m-separated from `Y` (given the empty set):

```
> msep(a, "X", "Y", "Z")

[1] FALSE

> msep(a, "X", "Y")

[1] TRUE
```

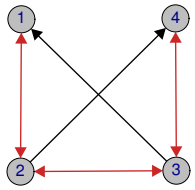
Verifying maximality

For many subclasses of graphs a missing edge corresponds to some independence statement, but for the more complex classes of mixed graphs this is not necessarily true. A graph where each of its missing edges is related to an independence statement is called a *maximal graph*. For a more detailed discussion on maximality of graphs and graph-theoretical conditions for maximal graphs, see Richardson and Spirtes (2002) and Sadeghi and Lauritzen (2012). Sadeghi and Lauritzen (2012) also gave an algorithm for generating maximal ribbonless graphs that induces the same independence structure as an input non-maximal ribbonless graph. This algorithm has been implemented in **ggm** as illustrated below.

Function for generating maximal graphs

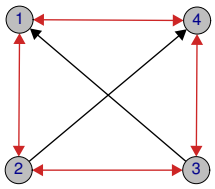
Given a non-maximal graph, we can obtain the adjacency matrix of a maximal graph that induces the same independence statements with the function `Max`. This function uses the algorithm by Sadeghi (2012b), which is an extension of the implicit algorithm presented in Richardson and Spirtes (2002). The related functions `MAG`, `MSG`, and `MRG`, are just handy wrappers to obtain maximal AGs, SGs and RGs, respectively. For example,

```
> H <- matrix(c(0  ,100,  1,  0,
                100,0  ,100,  0,
                0  ,100,  0,100,
                0,  1  ,100,  0), 4, 4)


> plotGraph(H)
```



is a non-maximal ancestral graph, with the missing edge between nodes 1 and 4 that is not associated with any independence statement. Its associated maximal graph is obtained by

```
> Max(H)


    1   2   3   4
1   0 100   0 100
2 100   0 100   1
3   1 100   0 100
4 100   0 100   0

> plotGraph(Max(H))
```



As the graph `H` is an ancestral graph (as may be verified by the function `isAG`), we obtain the same result with

```
> MAG(H)

    1   2   3   4
1   0 100   0 100
2 100   0 100   1
3   1 100   0 100
4 100   0 100   0
```

Verifying Markov equivalence

Two graphical models are said to be Markov equivalent when their associated graphs, although non-identical, imply the same independence structure, that is the same set of independence statements. Thus two Markov equivalent models cannot be distinguished on the basis of statistical tests of independence, even for arbitrary large samples. For instance, it is easy to verify that the two directed acyclic graphs models $X \prec\!\!\!- U \succ\!\!\!- Y$ and $X \prec\!\!\!- U \prec\!\!\!- Y$ both imply the same independence statements, and are, therefore, Markov equivalent.

Sometimes, we can check whether graphs of different types are Markov equivalent. For instance the DAG $X \succ\!\!\!- U \prec\!\!\!- Y$ is Markov equivalent to the bi-directed graph $X \prec\!\succ U \prec\!\succ Z$.

Markov equivalent models may be useful in applications because (a) they may suggest alternative interpretations of a given well-fitting model or (b) on the basis of the equivalence one can choose a simpler fitting algorithm. For instance, the previous bi-directed graph model may be fitted, using the Markov equivalent DAG, in terms of a sequence of univariate regressions.

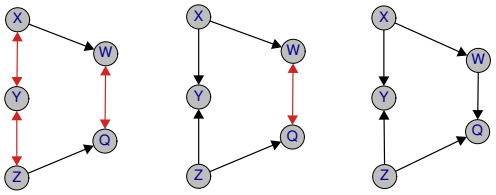
In the literature several problems related to Markov equivalences have been discussed. These include (a) verifying the Markov equivalence of given graphs, (b) presenting conditions under which a graph of a specific type can be Markov equivalent to a graph of another type, and (c) providing algorithms for generating Markov equivalent graphs of a certain type from a given graph.

Functions for testing Markov equivalences

The function `MarkEqRcg` tests whether two regression chain graphs are Markov equivalent. This function simply finds the skeleton and all unshielded collider V-configurations in both graphs and tests whether they are identical, see Wermuth and Sadeghi (2012). The arguments of this function are the two graphs `a` and `b` in one of the allowed forms. For example,

```
> H1 <- makeMG(dg = DAG(W ~ X, Q ~ Z),
               bg = UG(~ X*Y + Y*Z + W*Q))
> H2 <- makeMG(dg = DAG(W ~ X, Q ~ Z, Y ~ X + Z),
               bg = UG(~ W*Q))
> H3 <- DAG(W ~ X, Q ~ Z + W, Y ~ X + Z)
```

```
> plotGraph(H1); plotGraph(H2); plotGraph(H3)
```



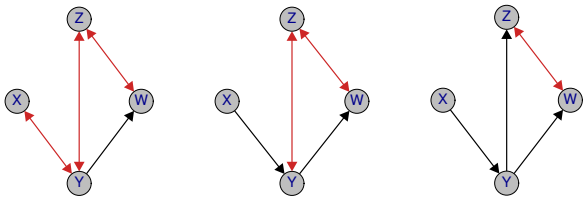
We can now verify Markov equivalence as follows

```
> MarkEqRcg(H1,H2)

[1] TRUE

> MarkEqRcg(H1,H3)

[1] FALSE

> MarkEqRcg(H2,H3)

[1] FALSE
```

To test Markov equivalence for maximal ancestral graphs the algorithm is much more computationally demanding (see Ali and Richardson (2004)) and, for this purpose, the function `MarkEqMag` has been provided. Of course, one can use this function for Markov equivalence of regression chain graphs (which are a subclass of maximal ancestral graphs). For example,

```
> A1 <- makeMG(dg = DG(W ~ Y),
               bg = UG(~ X*Y + Y*Z + Z*W))
> A2 <- makeMG(dg = DG(W ~ Y, Y ~ X),
               bg = UG(~ Y*Z + Z*W))
> A3 <- makeMG(dg = DG(W ~ Y, Y ~ X, Z ~ Y),
               bg = UG(~ Z*W))
```

```
> plotGraph(A1); plotGraph(A2); plotGraph(A3)
```

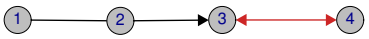



```
> MarkEqMag(H1,H2)

[1] TRUE

> MarkEqMag(H1,H3)

[1] FALSE

> MarkEqMag(H2,H3)

[1] FALSE
```

Functions for generating Markov equivalent graphs of a specific type

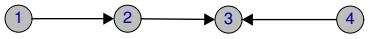
To obtain an alternative interpretation of an independence structure by using different graphical models, it is important to verify if a given graph is capable of being Markov equivalent to a graph of a specific class of graphs (such as DAGs, undirected graphs, or bidirected graphs), and if so, to obtain as a result such a graph. The functions `RepMarDAG`, `RepMarUG`, and `RepMarBG` do this for DAGs, undirected graphs, and bidirected graphs, respectively. For associated conditions and algorithms, see Sadeghi (2012b). For example, given the following graph

```
> H <- matrix(c( 0,10,  0,  0,
                10, 0,  0,  0,
                 0, 1,  0,100,
                 0, 0,100,  0), 4, 4)

> plotGraph(H)
```



we can see that it is Markov equivalent to a DAG, by

```
> RepMarDAG(H)

$verify
[1] TRUE

$amat
  1 2 3 4
1 0 1 0 0
2 0 0 1 0
3 0 0 0 0
4 0 0 1 0

> plotGraph(RepMarDAG(H))
```



On the other hand it is not Markov equivalent to an undirected graph or to a bidirected graph.

```
> RepMarUG(H)

$verify
[1] FALSE

$amat
[1] NA

> RepMarBG(H)

$verify
[1] FALSE

$amat
[1] NA
```

Acknowledgments

The authors are grateful to Steffen Lauritzen for helpful suggestions on codes and comments on an earlier version of the paper and to Nanny Wermuth, the editor, and referees for their insightful comments.

Bibliography

R. A. Ali and T. Richardson. Searching across Markov equivalence classes of maximal ancestral graphs. In *Proceedings of the Joint Statistical Meeting of the American Statistical Association*, Toronto, Canada, 2004. [p71]

D. R. Cox and N. Wermuth. *Multivariate Dependencies : models, analysis and interpretation*. Chapman & Hall, London, United Kingdom, 1996. [p66]

G. Csardi and T. Nepusz. The "igraph" software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.sf.net`. [p67]

C. Dethlefsen and S. Højsgaard. A common platform for graphical models in R: The gRbase package. *Journal of Statistical Software*, 14(17):1–12, 12 2005. ISSN 1548-7660. URL `http://www.jstatsoft.org/v14/i17`. [p67]

R. Gentleman, E. Whalen, W. Huber, and S. Falcon. *graph: A package to handle graph data structures*, 2012. R package version 1.36.0. [p67]

S. Højsgaard. Graphical independence networks with the gRain package for R. *Journal of Statistical Software*, 46(10):1–26, 2012. URL `http://www.jstatsoft.org/v46/i10/`. [p65]

S. Højsgaard, D. Edwards, and S. Lauritzen. *Graphical Models with R*. Springer-Verlag, Berlin-Heidelberg-New York, 2012. [p65, 67]

J. T. A. Koster. Marginalizing and conditioning in graphical models. *Bernoulli*, 8(6):817–840, 2002. [p66]

G. M. Marchetti. Independencies induced from a graphical Markov model after marginalization and conditioning: the R package ggm. *Journal of Statistical Software*, 15(6), 2006. [p66]

G. M. Marchetti, M. Drton, and K. Sadeghi. *ggm: A package for Graphical Markov Models*, 2012. URL `http://CRAN.R-project.org/package=ggm`. R package version 1.995-3. [p65, 66]

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1988. [p69]

T. S. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30(4):962–1030, 2002. [p66, 68, 69, 70]

K. Sadeghi. Stable mixed graphs. *Bernoulli*, to appear, 2012a. URL http://arxiv.org/abs/1110.4168. [p66, 68, 69]

K. Sadeghi. Markov equivalences for subclasses of mixed graphs. *Submitted*, 2012b. URL http://arxiv.org/abs/1110.4539. [p70, 72]

K. Sadeghi and S. L. Lauritzen. Markov properties for mixed graphs. *Bernoulli*, to appear, 2012. URL http://arxiv.org/abs/1109.5909. [p70]

N. Wermuth. Probability distributions with summary graph structure. *Bernoulli*, 17(3):845–879, 2011. [p66]

N. Wermuth and D. R. Cox. Distortion of effects caused by indirect confounding. *Biometrika*, 95:17–33, 2008. [p68]

N. Wermuth and K. Sadeghi. Sequences of regressions and their independences. *TEST*, 21(2):215–252 and 274–279, 2012. [p71]

Kayvan Sadeghi
Department of Statistics
University of Oxford
1 South Parks Road, OX1 3TG, Oxford
United Kingdom
sadeghi@stats.ox.ac.uk

Giovanni M. Marchetti
Dipartimento di Statistica "G. Parenti"
University of Florence
viale Morgagni, 59, 50134, Florence
Italy
giovanni.marchetti@ds.unifi.it

The State of Naming Conventions in R

by Rasmus Bååth

Abstract Most programming language communities have naming conventions that are generally agreed upon, that is, a set of rules that governs how functions and variables are named. This is not the case with R, and a review of unofficial style guides and naming convention usage on CRAN shows that a number of different naming conventions are currently in use. Some naming conventions are, however, more popular than others and as a newcomer to the R community or as a developer of a new package this could be useful to consider when choosing what naming convention to adopt.

Introduction

Most programming languages have official naming conventions, official in the sense that they are issued by the organization behind the language and accepted by its users. This is not the case with R. There exists the *R internals* document[1] which covers the coding standards of the R core team but it does not suggest any naming conventions. Incoherent naming of language entities is problematic in many ways. It makes it more difficult to guess the name of functions (for example, is it `as.date` or `as.Date`?). It also makes it more difficult to remember the names of parameters and functions. Two different functions can have the same name, where the only difference is the naming convention used. This is the case with `nrow` and `NROW` where both functions count the rows of a a data frame, but their behaviors differ slightly.

There exist many different naming conventions and below is a list of some of the most common. All are in use in the R community and the example names given are all from functions that are part of the **base** package. As whitespace cannot be part of a name, the main difference between the conventions is in how names consisting of multiple words are written.

alllowercase All letters are lower case and no separator is used in names consisting of multiple words as in `searchpaths` or `srcfilecopy`. This naming convention is common in MATLAB. Note that a single lowercase name, such as `mean`, conforms to all conventions but UpperCamelCase.

period.separated All letters are lower case and multiple words are separated by a period. This naming convention is unique to R and used in many core functions such as `as.numeric` or `read.table`.

underscore_separated All letters are lower case and multiple words are separated by an underscore as in `seq_along` or `package_version`. This naming convention is used for function and variable names in many languages including C++, Perl and Ruby.

lowerCamelCase Single word names consist of lower case letters and in names consisting of more than one word all, except the first word, are capitalized as in `colMeans` or `suppressPackageStartupMessage`. This naming convention is used, for example, for method names in Java and JavaScript.

UpperCamelCase All words are capitalized both when the name consists of a single word, as in `Vectorize`, or multiple words, as in `NextMethod`. This naming convention is used for class names in many languages including Java, Python and JavaScript.

If you are a newcomer to R or if you are developing a new package, how should you decide which naming convention to adopt? While there exist no official naming conventions there do exist a number of R style guides that include naming convention guidelines. Below is a non-exhaustive list of such guides.

- **Bioconductor's coding standards**
  http://wiki.fhcrc.org/bioc/Coding_Standards
- **Hadley Wickham's style guide**
  http://stat405.had.co.nz/r-style.html
- **Google's R style guide**
  http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html
- **Colin Gillespie's R style guide**
  http://csgillespie.wordpress.com/2010/11/23/r-style-guide/

Following a style guide will lead to good internal consistency in your code but you are still faced with the choice of naming conventions as there seems to be no consensus between style guides. The coding standards of the Bioconducor project recommend that both function and variable names are written in lowerCamelCase while Hadley Wickham's style guide recommends using underscore_separated names. Google's R style guide proposes UpperCamelCase for function names and period.separated variable names. Colin Gillespie's R style guide agrees with Google's on the the naming of functions but recommends underscore_separated variable names.

[1] http://cran.r-project.org/doc/manuals/R-ints.html

Naming conventions on CRAN

One thing to consider when choosing to adopt a naming convention is what conventions are already popular in the R community. For example, it is safe to say that it would be unconventional to release a package where function names are in all caps as in old FORTRAN. A good source of information regarding the current naming convention practices of the R community is the Comprehensive R Archive Network (CRAN). The function and parameter names used in CRAN packages should reflect the names R users are using, as CRAN is arguably the most common source for add-on packages.

In order to look into this I downloaded the documentation and the NAMESPACE files for all packages on CRAN [2]. The NAMESPACE files were used to extract function names and out of the 4108 packages on CRAN, function names from 2668 packages were retrieved. The reason why it was not possible to get function names from all packages is that while all CRAN packages now include a NAMESPACE file, not all NAMESPACE files explicitly export function names. S3 functions were converted not to include the class name, for example, `plot.myclass` just became `plot`. This was done in order to avoid inflating the number of period.separated function names. The documentation files were used to pick out the parameter names for all documented functions in order to get at what naming conventions are used when naming variables. In total 62,497 function names and 316,852 parameter names were retrieved.

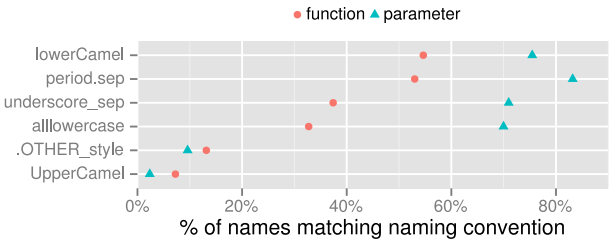


Figure 1: The percentage of function and parameter names from CRAN that matches the five naming conventions.

Figure 1 shows the percentage of function and parameter names that matches the five naming conventions, with lowerCamelCase and period.separated being the most common conventions. The impression, however, is that naming convention usage is quite heterogeneous as all of the five naming conventions seem to be used to some degree. Included in the figure is also the percentage of names that do not match any specified naming convention. These are labeled .OTHER_style. (Examples of such names would be `as.Date` and `Sys.setlocale`). Note that a name can match many naming conventions, especially all names that are alllowercase also match period.separated, underscore_separated and lowerCamelCase conventions. This explains why the parameter names match the top four naming conventions to a higher degree than the function names, as parameter names tend to be single letter words to a larger degree than function names (the single most common parameter name being `x`).

How common is it actually to mix naming conventions in the same package, given that there are many different naming conventions in use in the R community? Counting the minimum number of naming conventions required to cover all function names in each package on CRAN shows that while the largest group (43%) of packages stick to using one naming convention, 28% mix two naming conventions and 28% mix three or more.

Comparing the naming conventions advocated by the style guides with the situation on CRAN shows that some of the proposed naming conventions fit less well with the CRAN data. Both Google and Colin Gillespie propose using UpperCamelCase for function names, which seems to be far from the norm as only 7% of the function names on CRAN conform to this convention. Using underscore_separated names, as the style guide of Hadley Wickham proposes, is also relatively rare as compared to using lowerCamelCase or period.separated names. None of the style guides propose the naming convention that fits the CRAN data best, that is, to name functions using lowerCamelCase and variables using period.separated names. Although a case can be made for using the same naming convention for both variables and functions as, strictly speaking, functions are assigned to variables in R.

Both the CRAN data and the style guides show that there is no consensus regarding naming conventions in R and this it likely to continue as naming conventions, to a large degree, are a matter of taste and habit. If one believes that more homogeneous naming conventions are desirable it is a bit distressing that an entity as influential as Google issues naming convention guidelines that are not compatible with the current usage in the R community. What could help might be to raise awareness in the R community about naming conventions; writers of books and tutorials on R could make a difference here by treating naming conventions when introducing the R language. What is most important, however, is to keep a consistent naming convention style within your code base, whether you are working on a personal project or developing a package.

Rasmus Bååth
Lund University Cognitive Science
Lund University
Sweden
rasmus.baath@lucs.lu.se

[2] The files were retrieved from CRAN on 2012-11-13.

Changes in R

In version 2.15.2

by the R Core Team

CHANGES IN R VERSION 2.15.2

NEW FEATURES

- The `X11()` window gains an icon: the latter may be especially useful on Ubuntu's 'Unity' interface.

  The `WM_CLASS` should be set in circumstances where the Window Manager failed to make use of X11 resource settings.

  (Contributed by Philip Johnson.)

- The `"Date"` and `"POSIXt"` methods for `cut()` will accept an unsorted `breaks` argument (as the default method does, although this was undocumented). (Wish of PR#14961.)

- Reference class methods (in the **methods** package) that use other methods in an indirect way (e.g. by `sapply()`) must tell the code analysis to include that method. They can now do so by invoking `$usingMethods()`.

- More Polish translations are available: for the `RGui` menus and for several recommended packages.

- Multistratum MANOVA works. In fact, it seems to have done so for years in spite of the help page claiming it did not.

- `qqline()` has new optional arguments `distribution`, `probs` and `qtype`, following the example of **lattice**'s `panel.qqmathline()`.

- The handling of single quotes in the `en@quot` pseudo-language has been slightly improved. Double quotes are no longer converted.

- New functions `checkPoFiles()` and `checkPoFile()` have been added to the **tools** package to check for consistency of format strings in translation files.

- `model.matrix(~1,...)` now also contains the same rownames that less trivial formulae produce. (Wish of PR#14992, changes the output of several packages.)

- Misuse of `rep()` on undocumented types of objects (e.g. calls) is now reported as an error.

- The included LAPACK has been updated to 3.4.1, with some patches from the current SVN sources. (*Inter alia*, this resolves PR#14692.)

- `file.copy(recursive = TRUE)` has some additional checks on user error leading to attempted infinite recursion (and on some platforms to crashing R).

- PCRE has been updated to version 8.31, a bug-fix release.

- The included version of `liblzma` has been updated to version 5.0.4, a minor bug-fix release.

- New function `.bincode()`, a 'bare-bones' version of `cut.default(labels = FALSE)` for use in packages with `image()` methods.

- The HTML manuals now use directional single quotes.

- `maintainer()` now converts embedded new lines to spaces. It no longer gives a non-obvious error for non-installed packages.

- The `X11()` device has some protection against being used with forked processes *via* package **parallel**.

- Setting the environment variable `R_OSX_VALGRIND` (to any value) allows R to be run under `valgrind` on Mac OS 10.6 and 10.7 (`valgrind` currently has very limited support for 10.8), provided `system()` is not used (directly or indirectly). This should not be needed for `valgrind` >= 3.8.1.

- The `"model.frame"` method for `lm()` uses `xlevels`: this is safer if `data` was supplied or `model = FALSE` was used and the levels of factors used in the fit had been re-ordered since fitting.

  Similarly, `model.frame(fm,data=<data>)` copies across the variables used for safe prediction from the fit.

- Functions such as `parLapply()` in package **parallel** can make use of a default cluster if one is set. (Reported by Martin Morgan.)

- `chol(pivot = TRUE,LINPACK = FALSE)` is now available using LAPACK 3.2 subroutine `DPSTRF`.

- The functions `.C()`, `.Call()`, `.External()` and `.Fortran()` now check that they are called with an unnamed first argument: the formal arguments were changed from `name=` to `.NAME=` in R 2.13.0, but some packages were still using the old name. This is currently a warning, but will be an error in future.

- `step()` no longer tries to improve a model with AIC of `-Inf` (a perfect fit).

- `spline()` and `splinefun()` gain a new method `"hyman"`, an implementation of Hyman's method of constructing monotonic interpolation splines. (Based on contributions of Simon Wood and Rob Hyndman.)
- On Windows, the C stack size has been increased to 64MB (it has been 10MB since the days of 32MB RAM systems).

PERFORMANCE IMPROVEMENTS

- `array()` is now implemented in C code (for speed) when `data` is atomic or an unclassed list (so it is known that `as.vector(data)` will have no class to be used by `rep()`).
- `rep()` is faster and uses less memory, substantially so in some common cases (e.g. if `times` is of length one or `length.out` is given, and `each = 1`).
- `findInterval()`, `tabulate()`, `cut()`, `hist()` and `image.default()` all use `.Call()` and are more efficient.
- `duplicated()`, `unique()` and similar now support vectors of lengths above 2^{29} on 64-bit platforms.
- Omitting `PACKAGE` in `.C()` etc calls was supposed to make use of the DLL from the namespace within which the enclosing function was defined. It was less successful in doing so than it might be, and gave no indication it had failed.

  A new search strategy is very successful and gives a warning when it fails. In most cases this is because the entry point is not actually provided by that package (and so `PACKAGE` should be used to indicate which package is intended) but in some the namespace does not have a DLL specified by a `useDynLib()` directive so `PACKAGE` is required.

UTILITIES

- `R CMD check` now checks if a package can be loaded by `library(pkgname,lib.loc = "somewhere")` without being on the library search path (unless it is already installed in `.Library`, when it always will be).
- `R CMD check --as-cran` notes 'hidden' files and directories (with names starting with a dot) that are not needed for the operation of `R CMD INSTALL` or `R CMD build`: such files should be excluded from the published tarball.
- `R CMD check` (if checking subdirectories) checks that the R code in any demos is ASCII and can be parsed, and warns if not.
- When `R CMD Rd2pdf` is used with 'inputenx.sty', it allows further characters (mainly for Eastern European languages) by including 'ix-utf8enc.dfu' (if available). (Wish of PR#14989.)
- `R CMD build` now omits several types of hidden files/directories, including 'inst/doc/.Rinstignore', 'vignettes/.Rinstignore', ('.Rinstignore' should be at top level), '.deps' under 'src', '.Renviron', '.Rprofile', '.Rproj.user', '.backups', '.cvsignore', '.cproject', '.directory', '.dropbox', '.exrc', '.gdb.history', '.gitattributes', '.gitignore', '.gitmodules', '.hgignore', '.hgtags', '.htaccess', '.latex2html-init', '.project', '.seed', '.settings', '.tm_properties' and various leftovers.
- `R CMD check` now checks for `.C()`, `.Call()`, `.External()` and `.Fortran()` calls in other packages, and gives a warning on those found from R itself (which are not part of the API and change without notice: many will be changed for R 2.16.0).

C-LEVEL FACILITIES

- The limit for `R_alloc` on 64-bit platforms has been raised to just under 32GB (from just under 16GB).
- The misuse of `.C("name",...,PACKAGE = foo)` where `foo` is an arbitrary R object is now an error.

  The misuse `.C("name",...,PACKAGE = "")` is now warned about in `R CMD check`, and will be an error in future.

DEPRECATED AND DEFUNCT

- Use of `array()` with a 0-length `dim` argument is deprecated with a warning (and was contrary to the documentation).
- Use of `tapply()` with a 0-length `INDEX` list is deprecated with a warning.
- '`Translation`' packages are deprecated.
- Calling `rep()` or `rep.int()` on a pairlist is deprecated and will give a warning. In any case, `rep()` converted a pairlist to a list so you may as well do that explicitly.
- Entry point `rcont2` is no longer part of the API, and will move to package **stats** in R 2.16.0.
- The 'internal' graphics device invoked by `.Call("R_GD_nullDevice",package = "grDevices")` is about to be removed: use `pdf(file = NULL)` instead.

- `eigen(EISPACK = TRUE)`, `chol(pivot = FALSE,LINPACK = TRUE)`, `chol2inv(LINPACK = TRUE)`, `solve(LINPACK = TRUE)` and `svd(LINPACK = TRUE)` are deprecated and give a warning.

  They were provided for compatibility with R 1.7.0 (Mar 2003)!

- The 'internal function' `kappa.tri()` has been renamed to `.kappa_tri()` so it is not inadvertently called as a method for class `"tri"`.

- Functions `sessionData()` and `browseAll()` in package **methods** are on a help page describing them as 'deprecated' and are now formally deprecated.

PACKAGE INSTALLATION

- For a Windows or Mac OS X binary package install, `install.packages()` will check if a source package is available on the same repositories, and report if it is a later version or there is a source package but no binary package available.

  This check can be suppressed: see the help page.

- `install.packages(type = "both")` has been enhanced. In interactive use it will ask whether to choose the source version of a package if the binary version is older and contains compiled code, and also asks if source packages with no binary version should be installed).

INSTALLATION

- There is a new `configure` option '`--with-libtiff`' (mainly in case the system installation needs to be avoided).

- LAPACK 3.4.1 does use some Fortran 90 features, so `g77` no longer suffices.

- If an external LAPACK is used, it must be version 3.2 or later.

BUG FIXES

- On Windows, starting `Rterm` via `R.exe` caused Ctrl-C to misbehave. (PR#14948)

- The `tools::latexToUtf8()` function missed conversions that were contained within braces.

- Long timezone specifications (such as a file name preceded by `:`) could crash `as.POSIXlt`. (PR#14945)

- `R CMD build --resave-data` could fail if there was no 'data' directory but there was an 'R/sysdata.rda' file. (PR#14947)

- `is.na()` misbehaved on a 0-column data frame. (PR#14959)

- `anova.lmlist()` failed if `test` was supplied. (PR#14960)

  It was unable to compute Cp tests for object of class `"lm"` (it assumed class `"glm"`).

- The `formula` method for `sunflowerplot()` now allows `xlab` and `ylab` to be set. (Reported by Gerrit Eichner.)

- The `"POSIXt"` and `"Date"` methods for `hist()` could fail on Windows where adjustments to the right-hand boundary crossed a DST transition time.

- On Windows, the code in `as.POSIXct()` to handle incorrectly specified `isdst` fields might have resulted in `NA` being returned.

- `aov()` and `manova()` gave spurious warnings about a singular error model in the multiresponse case.

- In `ns()` and `bs()`, specifying `knots = NULL` is now equivalent to omitting it, also when `df` is specified. (PR#14970)

- `sprintf()` did not accept numbered arguments ending in zero. (PR#14975)

- `rWishart()` could overflow the C stack and maybe crash the R process for dimensions of several hundreds or more. (Reported by Michael Braun on R-sig-mac.)

- Base package vignettes (e.g. `vignette("Sweave")`) were not fully installed in builds of R from the tarball.

- `lchoose()` and `choose()` could overflow the C stack and crash R.

- When given a 0-byte file and asked to keep source references, `parse()` read input from `stdin()` instead.

- `pdf(compress = TRUE)` did not delete temporary files it created until the end of the R session. (PR#14991)

- `logLik()` did not detect the error of applying it to a multiple-response linear model. (PR#15000)

- `file.copy(recursive = TRUE)` did not always report `FALSE` for a failure two or more directories deep.

- `qgeom()` could return `-1` for extremely small `q`. (PR#14967.)

- `smooth.spline()` used `DUP = FALSE` which allowed its compiled C code to change the function: this was masked by the default byte-compilation. (PR#14965.)
- In Windows, the GUI preferences for foreground color were not always respected. (Reported by Benjamin Wells.)
- On OS X, the Quartz versions of the bitmap devices did not respect `antialias = "none"`. (PR#15006.)
- `unique()` and similar would infinite-loop if called on a vector of length $> 2^{29}$ (but reported that the vector was too long for 2^{30} or more).
- `parallel::stopCluster()` now works with MPI clusters without **snow** being on the search path.
- `terms.formula()` could exhaust the stack, and the stack check did not always catch this before the segfault. (PR#15013)
- `sort.list(method = "radix")` could give incorrect results on certain compilers (seen with `clang` on Mac OS 10.7 and `Xcode 4.4.1`).
- `backsolve(T,b)` gave incorrect results when `nrows(b) >ncols(T)` and `b` had more than one column.

  It could segfault or give nonsense if `k` was specified as more than `ncols(T)`.
- `smooth.spline()` did not check that a specified numeric `spar` was of length 1, and gave corrupt results if it was of length 0.
- Protection added to `do_system`. (PR#15025)
- Printing of vectors with names > 1000 characters now works correctly rather than truncating. (PR#15028)
- `qr()` for a complex matrix did not pivot the column names.
- `--with-blas='-framework vecLib'` now also works on OS X 10.8.
- `R CMD check` no longer fails with an error if a 'DESCRIPTION' file incorrectly contains a blank line. (Reported by Bill Dunlap.)
- `install.packages(type = "both")` could call `chooseCRANmirror()` twice.
- `lm.wfit()` could segfault in R 2.15.1 if all the weights were zero. (PR#15044)
- A malformed package name could cause `R CMD INSTALL` to write outside the target library.
- Some of the quality control functions (e.g. `tools::checkFF()`) were wrongly identifying the source of S4 methods in a package and so not checking them.
- The default type of display by `browseEnv()` when using `R.app` on Mac OS X has been incorrect for a long time.
- The implementation of `importMethodsFrom` in a `NAMESPACE` file could be confused and fail to find generics when importing from multiple packages (reported and fixed by Michael Lawrence).
- The detection of the C stack direction is better protected against compiler optimization. (PR#15011.)
- Long custom line types would sometimes segfault on the cairographics-based devices. (PR#15055.)
- `tools::checkPoFile()` unprotected too early in its C code and so segfaulted from time to time.
- The Fortran code underlying `nlminb()` could infinite-loop if any of the input functions returned `NA` or `NaN`. This is now an error for the gradient or Hessian, and a warning for the function (with the value replaced by `Inf`). (In part, PR#15052.)
- The code for creating `coerce()` methods could generate false notes about ambiguous selection; the notes have been suppressed for this function.
- `arima.sim()` could give too long an output in some corner cases (in part, PR#15068).
- `anova.glm()` with `test = "Rao"` didn't work when models included an offset. (Reported by Søren Feodor Nielsen.)
- `as.data.frame.matrix()` could return invalid data frame with no `row.names` attribute for 0-row matrix. (Reported by Hervé Pagès.)
- Compilation with the `vecLib` or `Accelerate` frameworks on OS X without using that also for LAPACK is more likely to be successful.

Changes on CRAN

2012-06-09 to 2012-11-28

by Kurt Hornik and Achim Zeileis

New packages in CRAN task views

Bayesian **BVS**, **Bayesthresh**, **MISA**, **bspmma**, **ggmcmc**, **growcurves**, **hbsae**, **pacbpred**, **rcppbugs**, **spTimer**.

ChemPhys **astroFns**, **cosmoFns**, **represent**.

ClinicalTrials **TrialSize***, **metaLik**.

Cluster **BayesLCA**, **HMMmix**, **Rmixmod***, **longclust**, **pgmm**, **teigen**.

Econometrics **AutoSEARCH**.

Finance **AutoSEARCH**, **TFX**, **pa**.

HighPerformanceComputing **HiPLARM**, **pbdBASE**, **pbdDMAT**, **pbdMPI**, **pbdSLAP**.

MachineLearning **C50**, **grpreg**.

MedicalImaging **mmand***.

OfficialStatistics **IC2**.

Optimization **CLSOCP**, **DWD**, **trustOptim**.

Phylogenetics **GUniFrac**, **HMPTrees**, **PVR**, **TESS**, **geomorph**, **phylotools**, **spider**.

Psychometrics **CopyDetect**, **lava**, **lava.tobit**, **rpf**, **semTools**, **simsem**.

SocialSciences **BradleyTerry2**.

Spatial **GriegSmith**, **OpenStreetMap**, **ggmap**, **osmar**, **plotKML**, **rasterVis**, **spacetime***, **spatialprobit**.

Survival **CR**, **FHtest**, **JMLSD**, **JMbayes**, **JPSurv**, **NPHMC**, **SGL**, **TBSSurvival**, **TPmsm**, **TestSurvRec**, **bpcp**, **complex.surv.dat.sim**, **compound.Cox**, **crrstep**, **fastcox**, **genSurv**, **jmec**, **joineR**, **lava.tobit**, **mets**, **survIDINRI**, **survSNP**.

TimeSeries **CommonTrend**, **ForeCA**, **Peak2Trough**, **TSA**, **WeightedPortTest**, **astsa**, **bfast**, **biwavelet**, **dlmodeler**, **x12**, **x12GUI**.

gR **QUIC**, **R2OpenBUGS**, **lcd**, **rjags**.

(* = core package)

New contributed packages

ACCLMA ACC & LMA Graph Plotting. Authors: Tal Carmi, Liat Gaziel.

ACD Categorical data analisys with complete or missing responses. Authors: Frederico Zanqueta Poleto, Julio da Mota Singer, Daniel Carlos Paulino, Fabio Mathias Correa and Enio Galinkin Jelihovschi.

AID Estimate Box-Cox Power Transformation Parameter. Authors: Osman Dag, Ozgur Asar, Ozlem Ilk.

AMAP.Seq Compare Gene Expressions from 2-Treatment RNA-Seq Experiments. Author: Yaqing Si.

AOfamilies Aranda-Ordaz (AO) transformation families. Authors: Hakim-Moulay Dehbi (with contributions from Mario Cortina-Borja and Marco Geraci).

APSIMBatch Analysis the output of Apsim software. Author: Bangyou Zheng.

Actigraphy Actigraphy Data Analysis. Authors: William Shannon, Tao Li, Hong Xian, Jia Wang, Elena Deych, Carlos Gonzalez.

ActuDistns Functions for actuarial scientists. Author: Saralees Nadarajah.

AdequacyModel Adequacy of models. Authors: Pedro Rafael Diniz Marinho, Cicero Rafael Barros Dias.

Agreement Statistical Tools for Measuring Agreement. Author: Yue Yu and Lawrence Lin.

AlleleRetain Allele Retention, Inbreeding, and Demography. Author: Emily Weiser.

AncestryMapper Ancestry Mapper. Authors: Tiago Magalhaes, Darren J. Fitzpatrick.

AssetPricing Optimal pricing of assets with fixed expiry date. Author: Rolf Turner.

AtmRay Acoustic Traveltime Calculations for 1-D Atmospheric Models. Author: Jake Anderson.

BACprior Sensitivity of the Bayesian Adjustment for Confounding (BAC) algorithm to the choice of hyperparameter omega. Author: Denis jf Talbot.

BADER Bayesian Analysis of Differential Expression in RNA Sequencing Data. Authors: Andreas Neudecker, Matthias Katzfuss.

BAEssd Bayesian Average Error approach to Sample Size Determination. Authors: Eric M. Reyes and Sujit K. Ghosh.

BDgraph Gaussian Graphical Model determination based on birth-death MCMC methodology. Authors: Abdolreza Mohammadi and Ernst Wit.

BGSIMD Block Gibbs Sampler with Incomplete Multinomial Distribution. Authors: Kwang Woo Ahn, Kung-Sik Chan.

BTYD Implementing Buy 'Til You Die Models. Authors: Lukasz Dziurzynski [aut], Edward Wadsworth [aut], Peter Fader [ctb], Elea McDonnell Feit [cre, ctb], Bruce Hardie [ctb], Arun Gopalakrishnan [ctb], Eric Schwartz [ctb], Yao Zhang [ctb].

BayesFactor Computation of Bayes factors for simple designs. Authors: Richard D. Morey, Jeffrey N. Rouder.

BayesNI Bayesian Testing Procedure for Noninferiority with Binary Endpoints. Authors: Sujit K Ghosh, Muhtarjan Osman.

Bayesthresh Bayesian thresholds mixed-effects models for categorical data. Authors: Fabio Mathias Correa and Julio Silvio de Souza Bueno Filho. In view: *Bayesian*.

BaylorEdPsych Baylor University Educational Psychology Quantitative Courses. Author: A. Alexander Beaujean.

Bclim Bayesian Palaeoclimate Reconstruction from Pollen. Authors: Andrew Parnell, Thinh Doan and James Sweeney.

BiGGR Creates an interface to BiGG database, provides a framework for simulation and produces flux graphs. Author: Anand K. Gavai.

BigTSP Top Scoring Pair based methods for classification. Authors: Xiaolin Yang,Han Liu.

Brq Bayesian analysis of quantile regression models. Author: Rahim Alhamzawi.

C50 C5.0 Decision Trees and Rule-Based Models. Authors: Max Kuhn, Steve Weston, Nathan Coulter. C code for C5.0 by R. Quinlan. In view: *MachineLearning*.

CARramps Reparameterized and marginalized posterior sampling for conditional autoregressive models. Authors: Kate Cowles and Stephen Bonett; with thanks to Juan Cervantes, Dong Liang, Alex Sawyer, and Michael Seedorff.

CBPS Covariate Balancing Propensity Score. Authors: Marc Ratkovic, Kosuke Imai, Christian Fong.

CCM Correlation Classification Method. Authors: Garrett M. Dancik and Yuanbin Ru.

CFL Compensatory Fuzzy Logic. Authors: Pablo Michel Marin Ortega, Kornelius Rohmeyer.

CGP Composite Gaussian Process models. Authors: Shan Ba and V. Roshan Joseph.

CHsharp Choi and Hall Clustering in 3d. Author: Douglas G. Woolford. In view: *Cluster*.

CMC Cronbach-Mesbah Curve. Authors: Michela Cameletti and Valeria Caviezel. In view: *Psychometrics*.

CR Power Calculation for Weighted Log-Rank Tests in Cure Rate Models. Authors: Emil A. Cornea, Bahjat F. Qaqish, and Joseph G. Ibrahim. In view: *Survival*.

CVST Fast Cross-Validation via Sequential Testing. Authors: Tammo Krueger, Mikio Braun.

CarbonEL Carbon Event Loop. Author: Simon Urbanek.

CensRegMod Fitting Normal and Student-t censored regression model. Authors: Monique Bettio Massuia, Larissa Avila Matos and Victor Lachos.

ChangeAnomalyDetection Change Anomaly Detection. Author: Yohei Sato.

CombMSC Combined Model Selection Criteria. Author: Andrew K. Smith.

CompLognormal Functions for actuarial scientists. Author: Saralees Nadarajah.

CompareTests Estimate diagnostic accuracy (sensitivity, specificity, etc) and agreement statistics when one test is conducted on only a subsample of specimens. Authors: Hormuzd A. Katki and David W. Edelstein.

CopulaRegression Bivariate Copula Based Regression Models. Authors: Nicole Kraemer, Daniel Silvestrini.

CopyDetect Computing Statistical Indices to Detect Answer Copying on Multiple-Choice Tests. Author: Cengiz Zopluoglu. In view: *Psychometrics*.

CorrBin Nonparametrics with clustered binary data. Author: Aniko Szabo.

DBKGrad Discrete Beta Kernel Graduation of mortality data. Authors: Angelo Mazza and Antonio Punzo.

DIME Differential Identification using Mixture Ensemble. Authors: Cenny Taslim, with contributions from Dustin Potter, Abbasali Khalili and Shili Lin.

Deducer Author: Ian Fellows with contributions from others (see documentation).

DeducerSurvival Add Survival Dialogue to Deducer. Authors: Matthew Ockendon, Paul Cool.

DeducerText Deducer GUI for Text Data. Authors: Alex Rickett and Ian Fellows, with contributions from Neal Fultz.

DiscreteInverseWeibull Discrete inverse Weibull distribution. Authors: Alessandro Barbiero, Riccardo Inchingolo.

DiscriMiner Tools of the Trade for Discriminant Analysis. Author: Gaston Sanchez.

DivMelt HRM Diversity Assay Analysis Tool. Authors: David Swan with contributions from Craig A Magaret and Matthew M Cousins.

DnE Distribution and Equation. Authors: Junyao Chen, Cuiyi He, Yuanrui Wu, Mengqing Sun.

DynClust Non-parametric denoising and clustering method of noisy images both indexed by time and space. Authors: Tiffany Lieury, Christophe Pouzat, Yves Rozenholc.

EBMAforecast Ensemble BMA Forecasting. Authors: Jacob M. Montgomery, Florian Hollenbach, and Michael D. Ward.

EBS Exact Bayesian Segmentation. Author: Alice Cleynen.

EDISON Software for network reconstruction and changepoint detection. Authors: Frank Dondelinger, Sophie Lebre.

EDanalysis Gene Enrichment Disequilibrium Analysis. Author: Yongshuai Jiang.

ENA Ensemble Network Aggregation. Author: Jeffrey D. Allen.

ETAS Modeling earthquake data using Epidemic Type Aftershock Sequence model. Authors: Abdollah Jalilian, based on Fortran code by Jiancang Zhuang.

EasyABC EasyABC: performing efficient approximate Bayesian computation sampling schemes. Authors: Franck Jabot, Thierry Faure, Nicolas Dumoullin.

EpiContactTrace Epidemiological tool for contact tracing. Author: Stefan Widgren, Maria Noremark.

ExPD2D Exact Computation of Bivariate Projection Depth Based on Fortran Code. Authors: Yijun Zuo, Xiangyang Ye.

ExPosition Exploratory analysis with the singular value decomposition. Authors: Derek Beaton, Cherise R. Chin Fatt, Herve Abdi.

FHtest Tests for right and interval-censored survival data based on the Fleming-Harrington class. Authors: Ramon Oller, Klaus Langohr. In view: *Survival*.

FMStable Finite Moment Stable Distributions. Author: Geoff Robinson.

FRACTION Numeric number into fraction. Author: OuYang Ming.

FRCC Fast Regularized Canonical Correlation Analysis. Author: Raul Cruz-Cano.

FWDselect Selecting variables in regression models. Authors: Marta Sestelo, Nora M. Villanueva, Javier Roca-Pardinas.

FactMixtAnalysis Factor Mixture Analysis with covariates. Author: Cinzia Viroli.

FinancialInstrument Financial Instrument Model Infrastructure for R. Authors: Peter Carl, Dirk Eddelbuettel, Jeffrey Ryan, Joshua Ulrich, Brian G. Peterson, Garrett See.

FindAllRoots Find all root(s) of the equation and Find root(s) of the equation by dichotomy. Author: Bingpei Wu, Jiajun He, Sijie Chen, Yangyang Liu.

FormalSeries Elementary arithemtic in formal series rings. Author: Tomasz Zmorzynski.

FourScores A game for two players. Author: Matthias Speidel.

GA Genetic Algorithms. Author: Luca Scrucca.

GA4Stratification A genetic algorithm approach to determine stratum boundaries and sample sizes of each stratum in stratified sampling. Authors: Sebnem Er, Timur Keskinturk, Charlie Daly.

GANPAdata The GANPA Datasets Package. Authors: Zhaoyuan Fang, Weidong Tian and Hongbin Ji.

GENEAread Reading Binary files. Author: Zhou Fang.

GISTools Some further GIS capabilities for R. Authors: Chris Brunsdon and Hongyan Chen.

GLDEX Fitting Single and Mixture of Generalised Lambda Distributions (RS and FMKL) using Various Methods. Authors: Steve Su, with contributions from: Diethelm Wuertz, Martin Maechler and Rmetrics core team members for low discrepancy algorithm, Juha Karvanen for

L moments codes, Robert King for gld C codes and starship codes, Benjamin Dean for corrections and input in ks.gof code and R core team for histsu function.

GOFSN Goodness-of-fit tests for the family of skew-normal models. Author: Veronica Paton Romero.

GPfit Gaussian Process Modeling. Authors: Blake MacDoanld, Hugh Chipman, Pritam Ranjan.

GWASExactHW Exact Hardy-Weinburg testing for Genome Wide Association Studies. Author: Ian Painter.

GeneF Generalized F-statistics. Author: Yinglei Lai.

Giza Constructing panels of population pyramid plots based on lattice. Author: Erich Striessnig.

HGNChelper Handy functions for working with HGNC gene symbols and Affymetrix probeset identifiers. Authors: Levi Waldron and Markus Riester.

HIBAG HLA Genotype Imputation with Attribute Bagging. Author: Xiuwen Zheng.

HMMmix HMM with mixture of gaussians as emission distribution. Authors: Stevenn Volant and Caroline Berard. In view: *Cluster*.

HPO.db A set of annotation maps describing the Human Phenotype Ontology. Author: Yue Deng.

HPbayes Heligman Pollard mortality model parameter estimation using Bayesian Melding with Incremental Mixture Importance Sampling. Author: David J Sharrow.

HW.pval Testing Hardy-Weinberg Equilibrium for Multiallelic Genes. Author: Shubhodeep Mukherji.

HandTill2001 Multiple Class Area under ROC Curve. Authors: Andreas Dominik Cullmann [aut, cre], Edgar Kublin [ctb].

HiPLARM High Performance Linear Algebra in R. Authors: Peter Nash and Vendel Szeremi. In view: *HighPerformanceComputing*.

Hotelling Hotelling's T-squared test and variants. Author: James Curran.

HyPhy Macroevolutionary phylogentic analysis of species trees and gene trees. Author: Nathaniel Malachi Hallinan.

HydroMe Estimation of Soil Hydraulic Parameters from Experimental Data. Author: Christian Thine Omuto. In view: *Environmetrics*.

ISDA.R interval symbolic data analysis for R. Authors: Ricardo Jorge de Almeida Queiroz Filho, Roberta Andrade de Araujo Fagundes.

ImpactIV Identifying Causal Effect for Multi-Component Intervention Using Instrumental Variable Method. Author: Peng Ding.

IndependenceTests Nonparametric tests of independence between random vectors. Authors: P Lafaye de Micheaux, M Bilodeau.

InfDim Infine-dimensional model (IDM) to analyse phenotypic variation in growth trajectories. Authors: Anna Kuparinen, Mats Bjorklund.

Interact Tests for marginal interactions in a 2 class response model. Authors: Noah Simon and Robert Tibshirani.

JASPAR R modules for JASPAR databases: a collection of transcription factor DNA-binding preferences, modeled as matrices. Author: Xiaobei Zhao.

JGL Performs the Joint Graphical Lasso for sparse inverse covariance estimation on multiple classes. Author: Patrick Danaher.

JMbayes Joint Modeling of Longitudinal and Time-to-Event Data under a Bayesian Approach. Author: Dimitris Rizopoulos. In view: *Survival*.

Julia Fractal Image Data Generator. Author: Mehmet Suzen.

Kpart Spline Fitting. Author: Eric Golinko.

LDdiag Link Function and Distribution Diagnostic Test for Social Science Researchers. Author: Yongmei Ni.

LICORS Light Cone Reconstruction of States — Predictive State Estimation From Spatio-Temporal Data. Author: Georg M. Goerg.

LIStest Longest Increasing Subsequence Independence Test. Authors: Jesus Garcia and Veronica Andrea Gonzalez Lopez.

LMest Fit Latent Markov models in basic versions. Author: Francesco Bartolucci.

LN3GV Author: Steve Lund.

LSC Local Statistical Complexity — Automatic Pattern Discovery in Spatio-Temporal Data. Author: Georg M. Goerg.

LTR Perform LTR analysis on microarray data. Author: Paul C. Boutros.

Laterality Authors: Borel A., Pouydebat E., Reghem E.

LifeTables Implement HMD model life table system. Authors: David J. Sharrow, GUI by Hana Sevcikova.

MATTOOLS Modern Calibration Functions for the Modern Analog Technique (MAT). Author: M. Sawada.

MBCluster.Seq Model-Based Clustering for RNA-seq Data. Author: Yaqing Si.

MBI (M)ultiple-site (B)iodiversity (I)ndices Calculator. Author: Youhua Chen.

MBmca Nucleic Acid Melting Curve Analysis on Microbead Surfaces with R. Author: Stefan Roediger.

MCUSUM Multivariate Cumulative Sum (MCUSUM) Control Chart. Author: Edgar Santos Fernandez.

MDSGUI A GUI for interactive MDS in R. Authors: Andrew Timm and Sugnet Gardner-Lubbe.

MESS Miscellaneous esoteric statistical scripts. Author: Claus Ekstrom.

MEWMA Multivariate Exponentially Weighted Moving Average (MEWMA) Control Chart. Author: Edgar Santos Fernandez.

MExPosition Multi-table ExPosition. Authors: Cherise R. Chin Fatt, Derek Beaton, Herve Abdi.

MMS Fixed effects Selection in Linear Mixed Models. Author: F. Rohart.

MPDiR Data sets and scripts for Modeling Psychophysical Data in R. Authors: Kenneth Knoblauch and Laurence T. Maloney.

MSQC Multivariate Statistical Quality Control. Author: Edgar Santos-Fernandez.

MTurkR Access to Amazon Mechanical Turk Requester API via R. Author: Thomas J. Leeper.

MVB Mutivariate Bernoulli log-linear model. Author: Bin Dai.

MeDiChI MeDiChI ChIP-chip deconvolution library. Author: David J Reiss.

Metrics Evaluation metrics for machine learning. Author: Ben Hamner.

MindOnStats Data sets included in Utts and Heckard's Mind on Statistics. Author: Jonathan Godfrey.

Miney Implementation of the Well-Known Game to Clear Bombs from a Given Field (Matrix). Author: Roland Rau.

MitISEM Mixture of Student t Distributions using Importance Sampling and Expectation Maximization. Authors: N. Basturk, L.F. Hoogerheide, A. Opschoor, H.K. van Dijk.

MultiChIPmixHMM Author: Caroline Berard.

MultiLCIRT Multidimensional latent class Item Response Theory models. Authors: Francesco Bartolucci, Silvia Bacci, Michela Gnaldi.

NLRoot Searching for the root of equation. Authors: Zheng Sengui, Lu Xufen, Hou Qiongchen, Zheng Jianhui.

NMRS NMR Spectroscopy. Author: Jose L. Izquierdo. In view: *ChemPhys*.

NPHMC Sample Size Calculation for the Proportional Hazards Cure Model. Authors: Chao Cai, Songfeng Wang, Wenbin Lu, Jiajia Zhang. In view: *Survival*.

NPMPM Tertiary probabilistic model in predictive microbiology for use in food manufacture. Author: Nadine Schoene.

NScluster Simulation and Estimation of the Neyman-Scott Type Spatial Cluster Models. Authors: The Institute of Statistical Mathematics, based on the program by Ushio Tanaka.

NonpModelCheck Model Checking and Variable Selection in Nonparametric Regression. Author: Adriano Zanin Zambom.

OPE Outer-product emulator. Author: Jonathan Rougier.

OPI Open Perimetry Interface. Author: Andrew Turpin.

ORDER2PARENT Estimate parent distributions with data of several order statistics. Author: Cheng Chou.

PAS Polygenic Analysis System (PAS). Author: Zhiqiu Hu; Shizhong Xu; Zhiquan Wang; Rongcai Yang.

PCS Calculate the probability of correct selection (PCS). Author: Jason Wilson.

PDSCE Positive definite sparse covariance estimators. Author: Adam J. Rothman.

PF Functions related to prevented fraction. Author: Dave Siev.

PKI Public Key Infrastucture for R based on the X.509 standard. Author: Simon Urbanek.

PKmodelFinder Software for Pharmacokinetic model. Authors: Eun-Kyung Lee, Gyujeong Noh, Hyeong-Seok Lim.

PLIS Multiplicity control using Pooled LIS statistic. Author: Zhi Wei, Wenguang Sun.

POET Principal Orthogonal ComplEment Thresholding (POET) method. Authors: Jianqing Fan, Yuan Liao, Martina Mincheva.

PPtree Projection pursuit classification tree. Authors: Eun-Kyung Lee, Yoondong Lee.

PRISMA Protocol Inspection and State Machine Analysis. Authors: Tammo Krueger, Nicole Kraemer.

PSCN Parent Specific DNA Copy Number Estimation. Authors: Hao Chen, Haipeng Xing, and Nancy R. Zhang.

PVAClone Population Viability Analysis with Data Cloning. Authors: Khurram Nadeem, Peter Solymos.

PVR Computes Phylogenetic eigenVectors Regression and Phylogentic Signal-Representation curve (with null and Brownian expectancies). Authors: Thiago Santos, Jose Alexandre Diniz-Filho, Thiago Rangel and Luis Mauricio Bini. In view: *Phylogenetics*.

Peaks Author: Miroslav Morhac. In view: *ChemPhys*.

PenLNM Group l1 penalized logistic normal multinomial (LNM) regression model. Author: Fan Xia.

PermAlgo Permutational algorithm to simulate survival data. Authors: Marie-Pierre Sylvestre, Thad Edens, Todd MacKenzie, Michal Abrahamowicz. In view: *Survival*.

PlayerRatings Dynamic Updating Methods For Player Ratings Estimation. Authors: Alec Stephenson and Jeff Sonas.

PopGenKit Useful functions for (batch) file conversion and data resampling in microsatellite datasets. Author: Sebastien Rioux Paquette.

PopGenReport PopGen: A simple way to analyse and visualize population genetic data. Author: Aaron Adamack, Bernd Gruber.

ProgGUIinR Support package for "Programming Graphical User Interfaces in R". Authors: Michael Lawrence and John Verzani.

QLSpline Author: Steve Lund.

R0 Estimation of R0 and real-time reproduction number from epidemics. Authors: Pierre-Yves Boelle, Thomas Obadia.

R1magic Compressive Sampling: Sparse signal recovery utilities. Author: Mehmet Suzen.

R2MLwiN Running MLwiN from within R. Authors: Zhengzheng Zhang, Chris Charlton, Richard Parker, George Leckie, William Browne.

RAMpath Authors: Zhiyong Zhang, Jack McArdle, Aki Hamagami, & Kevin Grimm.

RAppArmor Author: Jeroen Ooms.

RAtmosphere Standard Atmosperic profiles. Author: Gionata Biavati.

RDF RDF reading and writing. Author: Willem Robert van Hage.

REPPlab R interface to EPP-lab, a Java program for exploratory projection pursuit. Authors: Daniel Fischer, Alain Berro, Klaus Nordhausen, Anne Ruiz-Gazen.

RGCCA Regularized Generalized Canonical Correlation Analysis. Author: Arthur Tenenhaus.

RHT Regularized Hotelling's T-square Test for Pathway (Gene Set) Analysis. Authors: Lin S. Chen and Pei Wang.

ROCwoGS Non-parametric estimation of ROC curves without Gold Standard Test. Author: Chong Wang.

RPCLR RPCLR (Random-Penalized Conditional Logistic Regression). Author: Raji Balasubramanian.

RSclient Client for Rserve. Author: Simon Urbanek.

RTriangle A 2D Quality Mesh Generator and Delaunay Triangulator. Authors: Jonathan Shewchuk, David C. Sterratt.

RVideoPoker Play Video Poker with R. Authors: Roland Rau; cards were created by Byron Knoll.

RVtests Rare Variant Tests Using Multiple Regression Methods. Authors: C. Xu, C. M. Greenwood.

RandForestGUI Authors: Rory Michelland, Genevieve Grundmann.

Rarity Calculation of rarity indices for species and assemblages of species. Author: Boris Leroy.

Rchemcpp R interface for the ChemCpp library. Authors: Michael Mahr, Guenter Klambauer.

RcmdrPlugin.EACSPIR Plugin de R-Commander para el manual EACSPIR. Authors: Maribel Peró, David Leiva, Joan Guàrdia, Antonio Solanas.

RcmdrPlugin.MPAStats R Commander Plug-in for MPA Statistics. Author: Andrew Heiss.

RcmdrPlugin.PT Some discrete exponential dispersion models: Poisson-Tweedie. Authors: David Pechel Cactcha, Laure Pauline Fotso and Celestin C Kokonendji.

RcmdrPlugin.SLC SLC **Rcmdr** Plug-in. Authors: Antonio Solanas, Rumen Manolov.

RcmdrPlugin.StatisticalURV Statistical URV **Rcmdr** Plug-In. Author: Daniela Vicente.

RcmdrPlugin.TextMining R commander plugin for **tm** package. Author: Dzemil Lusija. In view: *NaturalLanguageProcessing*.

RcmdrPlugin.coin Rcmdr coin Plug-In. Author: Daniel-Corneliu Leucuta.

RcmdrPlugin.depthTools R commander Depth Tools Plug-In. Authors: Sara Lopez-Pintado and Aurora Torrente.

RcppCNPy Rcpp bindings for NumPy files. Author: Dirk Eddelbuettel.

RcppOctave Seamless Interface to Octave and Matlab. Author: Renaud Gaujoux.

Rdpack Update and manipulate Rd documentation objects. Author: Georgi N. Boshnakov.

Rearrangement Monotonize point and interval functional estimates by rearrangement. Authors: Wesley Graybill, Mingli Chen, Victor Chernozhukov, Ivan Fernandez-Val, Alfred Galichon.

Records Record Values and Record Times. Author: Magdalena Chrapek.

ReliabilityTheory Tools for structural reliability analysis. Author: Louis Aslett.

Ridit Ridit Analysis (An extension of the Kruskal-Wallis Test.). Author: SeyedMahmood TaghaviShahri.

Rivivc In vitro in vivo correlation linear level A. Authors: Aleksander Mendyk, Sebastian Polak.

Rsundials Suite of Nonlinear Differential Algebraic Equations Solvers in R. Author: Selwyn-Lloyd McPherson.

Rttf2pt1 Package for ttf2pt1 program. Authors: Winston Chang, Andrew Weeks, Frank M. Siegert, Mark Heath, Thomas Henlick, Sergey Babkin, Turgut Uyar, Rihardas Hepas, Szalay Tamas, Johan Vromans, Petr Titera, Lei Wang, Chen Xiangyang, Zvezdan Petkovic, Rigel, I. Lee Hetherington.

Rvelslant Downhole Seismic Analysis in R. Authors: Original method by Dave Boore, R port and some additions by Eric M. Thompson.

SAM Sparse Additive Machine. Authors: Tuo Zhao, Xingguo Li, Han Liu, Lie Wang, Kathryn Roeder.

SECP Statistical Estimation of Cluster Parameters (SECP). Author: Pavel V. Moskalev.

SEERaBomb SEER Setup and Use with A-Bomb Data. Author: Tomas Radivoyevitch.

SMR Studentized Midrange Distribution. Authors: Ben Deivide de Oliveira Batista, Daniel Furtado Ferreira.

STARSEQ Secondary Trait Association analysis for Rare variants via SEQuence data. Author: Dajiang Liu.

SWATmodel A multi-OS implementation of the TAMU SWAT model. Authors: Fuka, DR, Walter, MT, and Easton, ZM.

ScreenClean Screen and clean variable selection procedures. Authors: Pengsheng Ji, Jiashun Jin, Qi Zhang.

Segmentor3IsBack A Fast Segmentation Algorithm. Authors: Alice Cleynen, Guillem Rigaill, Michel Koskas.

Sejong KoNLP static dictionaries and Sejong project resources. Author: Heewon Jeon.

SensitivityCaseControl Sensitivity Analysis for Case-Control Studies. Author: Dylan Small.

SeqGrapheR Simple GUI for graph based visualization of cluster of DNA sequence reads. Author: Petr Novak.

SimCorMultRes Simulates Correlated Multinomial Responses. Author: Anestis Touloumis.

Simpsons Detecting Simpson's Paradox. Author: Rogier Kievit, Sacha Epskamp.

SimultAnR Correspondence and Simultaneous Analysis. Authors: Amaya Zarraga, Beatriz Goitisolo.

Sleuth3 Data sets from Ramsey and Schafer's "Statistical Sleuth (3rd ed)". Authors: Original by F.L. Ramsey and D.W. Schafer, modifications by Daniel W. Schafer, Jeannie Sifneos and Berwin A. Turlach.

SmarterPoland A set of tools developed by the Foundation SmarterPoland.pl. Author: Przemyslaw Biecek.

SpatialPack Analysis of spatial data. Authors: Felipe Osorio, Ronny Vallejos, and Francisco Cuevas.

Stem Spatio-temporal models in R. Author: Michela Cameletti. In view: *Spatial*.

StressStrength Computation and estimation of reliability of stress-strength models. Authors: Alessandro Barbiero, Riccardo Inchingolo.

TAHMMAnnot Mixture model approach to compare two samples of Tiling Array data. Author: Caroline Berard.

TANOVA Time Course Analysis of Variance for Microarray. Authors: Baiyu Zhou and Weihong Xu.

TBSSurvival TBS Model R package. Authors: Adriano Polpo, Cassio de Campos, D. Sinha, Stuart Lipsitz, Jianchang Lin. In view: *Survival*.

TERAplusB Test for A+B Traditional Escalation Rule. Author: Eun-Kyung Lee.

TESS Fast simulation of reconstructed phylogenetic trees under time-dependent birth-death processes. Author: Sebastian Hoehna. In view: *Phylogenetics*.

TExPosition Two-table ExPosition. Authors: Derek Beaton, Jenny Rieck, Cherise R. Chin Fatt, Herve Abdi.

TFX R API to TrueFX(tm). Author: Garrett See. In view: *Finance*.

TPmsm Estimation of transitions probabilities in multistate models. Authors: Artur Agostinho Araujo, Javier Roca-Pardinas and Luis Meira-Machado. In view: *Survival*.

TRIANGG General discrete triangular distribution. Authors: Tristan Senga Kiessé, Francial G. Libengué, Silvio S. Zocchi, Célestin C. Kokonendji.

TSEN Two-dimensional peak sentinel tool for GC x GC-HRTOFMS. Author: Yasuyuki Zushi.

TSPC Prediction using time-course gene expression. Author: Yuping Zhang.

TScompare TSdbi Comparison. Author: Paul Gilbert.

TSdata TSdbi Illustration. Author: Paul Gilbert.

ThreeWay Three-way component analysis. Authors: Maria Antonietta Del Ferraro, Henk A.L. Kiers, Paolo Giordani.

TrialSize Author: Ed Zhang. In view: *ClinicalTrials*.

Tsphere Transposable Sphering for Large-Scale Inference with Correlated Data. Author: Genevera I. Allen.

TukeyC Conventional Tukey Test. Authors: Jose Claudio Faria, Enio Jelihovschi and Ivan Bezerra Allaman.

TwoCop Nonparametric test of equality between two copulas. Authors: Bruno Remillard and Jean-Francois Plante.

UScensus2000blkgrp US Census 2000 Block Group Shapefiles and Additional Demographic Data. Author: Zack W. Almquist. In view: *Spatial*.

VBMA4hmm Variational Bayesian Markov Model for hidden markov model. Author: Stevenn Volant.

VDA Authors: Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu.

VIM Visualization and Imputation of Missing Values. Authors: Matthias Templ, Andreas Alfons, Alexander Kowarik, Bernd Prantner. In views: *Multivariate*, *OfficialStatistics*.

VineCopula Statistical inference of vine copulas. Authors: Ulf Schepsmeier, Jakob Stoeber, Eike Christian Brechmann.

VizCompX Visualisation of Computer Models. Author: Neil Diamond.

W2CWM2C A set of functions to produce new graphical tools for wavelet correlation (bivariate and multivariate cases) using some routines from the **waveslim** and **wavemulcor** packages. Author: Josue Moises Polanco-Martinez.

WCQ Detection of QTL effects in a small mapping population. Author: Jan Michael Yap.

WMDB Discriminant Analysis Methods by Weight Mahalanobis Distance and bayes. Author: Bingpei Wu.

WaveCD Wavelet change point detection for array CGH data. Author: M. Shahidul Islam.

WaveletCo Wavelet Coherence Analysis. Author: Huidong Tian; Bernard Cazelles.

ZeligChoice Zelig Choice Models. Authors: Matt Owen, Kosuke Imai, Olivia Lau and Gary King.

ZeligGAM Genereal Additive Models for **Zelig**. Authors: Matthew Owen, Skyler Cranmer, Olivia Lau, Kosuke Imai and Gary King.

ZeligMultilevel Multilevel Regressions for **Zelig**. Authors: Matthew Owen, Ferdinand Alimadhi, Delia Bailey.

adaptsmoFMRI Adaptive Smoothing of FMRI Data. Author: Max Hughes.

afex Analysis of Factorial Experiments. Author: Henrik Singmann.

aftgee Accelerated Failure Time Model with Generalized Estimating Equations. Authors: Sy Han (Steven) Chiou, Sangwook Kang, Jun Yan.

agRee Various Methods for Measuring Agreement. Author: Dai Feng.

ageprior Prior distributions for molecular dating. Author: Michael Matschiner.

aggrisk Estimate individual level risk using individual case data and spatially aggregated control data. Authors: Michelle Stanton, Yongtao Guan.

allanvar Allan Variance Analysis. Author: Javier Hidalgo Carrio.

amen Additive and multiplicative effects modeling of networks and relational data. Authors: Peter Hoff, Bailey Fosdick, Alex Volfovsky, Kate Stovel.

anaglyph 3D Anaglyph Plots. Author: Jonathan Lee.

andrews Andrews curves. Author: Jaroslav Myslivec.

antitrust Authors: Michael Sandfort and Charles Taragin.

aqr Interface methods to access use an ActiveQuant Master Server. Author: Ulrich Staudinger.

asd Simulations for adaptive seamless designs. Author: Nick parsons. In views: *ClinicalTrials*, *ExperimentalDesign*.

astroFns Miscellaneous astronomy functions, utilities, and data. Author: Andrew Harris. In view: *ChemPhys*.

astsa Applied Statistical Time Series Analysis. Author: David Stoffer. In view: *TimeSeries*.

attfad Evaluation and comparison of expression data and GRNs. Author: Robert Maier.

audit Bounds for Accounting Populations. Author: Glen Meeden.

autopls pls regression with backward selection of predictors. Authors: Sebastian Schmidtlein, with contributions from Carsten Oldenburg and Hannes Feilhauer.

bReeze Functions for wind resource assessment. Authors: Christian Graul and Carsten Poppinga.

base64enc Tools for base64 encoding. Author: Simon Urbanek.

batade HTML reports and so on. Author: Ichikawa Daisuke.

batchmeans Consistent Batch Means Estimation of Monte Carlo Standard Errors. Authors: Murali Haran and John Hughes.

bayesGDS Functions to implement Generalized Direct Sampling. Author: Michael Braun.

bayespref Hierarchical Bayesian analysis of ecological count data. Authors: Zachariah Gompert and James A. Fordyce.

bc3net Authors: Ricardo de Matos Simoes and Frank Emmert-Streib.

bdpv Inference and design for predictive values in binary diagnostic tests. Author: Frank Schaarschmidt.

beadarrayFilter Bead filtering for Illumina bead arrays. Authors: Anyiawung Chiara Forcheh, Geert Verbeke, Adetayo Kasim, Dan Lin, Ziv Shkedy, Willem Talloen, Hinrich WH Gohlmann, Lieven Clement.

betafam Detecting rare variants for quantitative traits using nuclear families. Author: Wei Guo.

biasbetareg Bias correction of the parameter estimates of the beta regression model. Author: Luana Cecilia Meireles.

bigmemory.sri A shared resource interface for Bigmemory Project packages. Author: Michael J. Kane.

bimetallic Power for SNP analyses using silver standard cases. Author: Andrew McDavid.

binhf Haar-Fisz functions for binomial data. Author: Matt Nunes.

binomialcftp Generates binomial random numbers via the coupling from the past algorithm. Author: Francisco Juretig.

binseqtest Exact Binary Sequential Designs and Analysis. Authors: Jenn Kirk, Mike Fay.

biomod2 Ensemble platform for species distribution modeling. Authors: Wilfried Thuiller, Damien Georges and Robin Engler.

bise Auxiliary functions for phenological data analysis. Author: Daniel Doktor.

biseVec Auxiliary functions for phenological data analysis. Author: Maximilian Lange.

bisoreg Bayesian Isotonic Regression with Bernstein Polynomials. Author: S. McKay Curtis. In view: *Bayesian*.

bivarRIpower Sample size calculations for bivariate longitudinal data. Authors: W. Scott Comulada and Robert E. Weiss.

biwt Functions to compute the biweight mean vector and covariance & correlation matrices. Author: Jo Hardin.

blockcluster Co-Clustering for Binary, Contingency and Continuous data-sets. Authors: Parmeet Bhatia, Serge Iovleff and Gerard Goavert, with contributions from Christophe Biernacki and Gilles Celeux.

bmk MCMC diagnostics. Authors: Matthew Krachey and Edward L. Boone.

boostSeq Optimized GWAS cohort subset selection for resequencing studies. Author: Milan Hiersche.

bootES Bootstrap Effect Sizes. Authors: Daniel Gerlanc and Kris Kirby.

bootfs Use multiple feature selection algorithms to derive robust feature sets for two class classification problems. Author: Christian Bender.

bpcp Beta Product Confidence Procedure for Right Censored Data. Author: Michael Fay. In view: *Survival*.

breakage SICM pipette tip geometry estimation. Author: Matthew Caldwell.

broman Karl Broman's R code. Author: Karl W Broman.

bspmma Bayesian Semiparametric Models for Meta-Analysis. Author: Deborah Burr. In view: *Bayesian*.

bursts Markov model for bursty behavior in streams. Author: Jeff Binder.

bvenn A Simple alternative to proportional Venn diagrams. Author: Raivo Kolde.

cancerTiming Estimation of temporal ordering of cancer abnormalities. Author: Elizabeth Purdom.

capme Covariate Adjusted Precision Matrix Estimation. Authors: T. Tony Cai, Hongzhe Li, Weidong Liu and Jichun Xie.

capwire Estimates population size from non-invasive sampling. Authors: Matthew W. Pennell and Craig R. Miller.

carcass Estimation of the number of fatalities from carcass searches. Authors: Fraenzi Korner-Nievergelt, Ivo Niermann, Oliver Behr, Robert Brinkmann, Pius Korner, Barbara Hellriegel, Manuela Huso.

caspar Clustered and sparse regression (CaSpaR). Author: Daniel Percival.

catIrt Simulating IRT-Based Computerized Adaptive Tests. Author: Steven W. Nydick.

ccChooser Developing core collections. Authors: Marcin Studnicki and Konrad Debski.

ccaPP (Robust) canonical correlation analysis via projection pursuit. Authors: Andreas Alfons [aut, cre], David Simcha [ctb].

cggd Continuous Generalized Gradient Descent. Authors: Cun-Hui Zhang and Ofer Melnik.

charlson Converts listwise icd9 data into comorbidity count and Charlson Index. Author: Vanessa Cox.

cheb Discrete Linear Chebyshev Approximation. Author: Jan de Leeuw.

cheddar Analysis and visualisation of ecological communities. Authors: Lawrence Hudson with contributions from Dan Reuman and Rob Emerson.

chords Estimation in respondent driven samples. Author: Jonathan Rosenblatt.

classify Classification Accuracy and Consistency under IRT models. Authors: Dr Chris Wheadon and Dr Ian Stockford.

clusterCrit Clustering Indices. Author: Bernard Desgraupes.

clustergas A hierarchical clustering method based on genetic algorithms. Authors: Jose A. Castellanos-Garzon, Fernando Diaz.

clusteval Evaluation of Clustering Algorithms. Author: John A. Ramey.

clusthaplo Authors: Damien Leroux, Brigitte Mangin, Sylvain Jasson, Abdelaziz Rahmani.

coalescentMCMC MCMC Algorithms for the Coalescent. Author: Emmanuel Paradis.

cocron Statistical comparisons of two or more alpha coefficients. Author: Birk Diedenhofen.

coenoflex Gradient-Based Coenospace Vegetation Simulator. Author: David W. Roberts.

coexist Species coexistence modeling and analysis. Author: Youhua Chen.

colcor Tests for column correlation in the presence of row correlation. Author: Omkar Muralidharan.

colortools Tools for colors in an HSV color model. Author: Gaston Sanchez.

commandr Command pattern in R. Author: Michael Lawrence.

comorbidities Categorizes ICD-9-CM codes based on published comorbidity indices. Author: Paul Gerrard.

compareODM Comparison of medical forms in CDISC ODM format. Author: Martin Dugas.

compoisson Conway-Maxwell-Poisson Distribution. Author: Jeffrey Dunn. In view: *Distributions*.

conics Plot Conics. Author: Bernard Desgraupes.

cosmoFns Functions for cosmological distances, times, luminosities, etc. Author: Andrew Harris. In view: *ChemPhys*.

cotrend Consistant Cotrend Rank Selection. Author: A. Christian Silva.

csSAM Cell-specific Significance Analysis of Microarrays. Authors: Shai Shen-Orr, Rob Tibshirani, Narasimhan Balasubramanian, David Wang.

cudia CUDIA Cross-level Imputation. Authors: Yubin Park and Joydeep Ghosh.

cvq2 Calculate the predictive squared correlation coefficient. Author: Torsten Thalheim.

cxxfunplus extend cxxfunction by saving the dynamic shared objects. Author: Jiqiang Guo.

daewr Design and Analysis of Experiments with R. Author: John Lawson.

datamart Unified access to various data sources. Author: Karsten Weinert.

datamerge Merging of overlapping and inconsistent data. Author: Christofer Backlin.

dbConnect Provides a graphical user interface to connect with databases that use MySQL. Authors: Dason Kurkiewicz, Heike Hofmann, Ulrike Genschel.

dbmss Distance-based measures of spatial structures. Authors: Eric Marcon, Gabriel Lang, Stephane Traissac, Florence Puech.

deamer Deconvolution density estimation with adaptive methods for a variable prone to measurement error. Authors: Julien Stirnemann, Adeline Samson, Fabienne Comte. Contribution from Claire Lacour.

degenes Detection of differentially expressed genes. Author: Klaus Jung.

depthTools Depth Tools. Authors: Sara Lopez-Pintado and Aurora Torrente.

dglars Differential Geometric LARS (dgLARS) method. Author: Luigi Augugliaro.

dgof Discrete Goodness-of-Fit Tests. Authors: Taylor B. Arnold, John W. Emerson, R Core Team and contributors worldwide.

dinamic DiNAMIC A Method To Analyze Recurrent DNA Copy Number Aberrations in Tumors. Authors: Vonn Walter, Andrew B. Nobel, and Fred A. Wright.

distfree.cr Distribution-free confidence region (distfree.cr). Authors: Zhiqiu Hu, Rong-cai Yang.

divagis Provides tools for quality checks of georeferenced plant species accessions. Author: Reinhard Simon.

diveRsity Genetic diversity partition statistics and Informative locus selection using Fst, Gst, Dest(Jost Chao) G'st and In. Author: Kevin Keenan.

dna Differential Network Analysis. Authors: Ryan Gill, Somnath Datta, Susmita Datta.

downloader Downloading files over https. Author: Winston Chang.

dpa Dynamic Path Approach. Author: Emile Chappin.

dpglasso Primal Graphical Lasso. Authors: Rahul Mazumder and Trevor Hastie.

drawExpression Visualising R syntax through graphics. Author: Sylvain Loiseau.

ds Descriptive Statistics. Author: Emmanuel Arnhold.

dsample Discretization-based Direct Random Sample Generation. Authors: Liqun Wang and Chel Hee Lee.

dsm Density surface modelling (dsm) of distance sampling data. Authors: David L. Miller, Eric Rexstad, Louise Burt, Mark V. Bravington, Sharon Hedley.

dynCorr Dynamic Correlation. Authors: Joel Dubin, Dandi Qiao, Hans-Georg Mueller.

easi EASI Demand System Estimation. Authors: Stephane Hoareau, Guy Lacroix, Mirella Hoareau, Luca Tiberti.

easyanova Analysis of variance and other important complementary analyzes. Author: Emmanuel Arnhold.

edesign Maximum entropy sampling. Author: Claudia Gebhardt.

eeptools Convenience functions for education data. Author: Jared E. Knowles.

eigeninv Generates (dense) matrices that have a given set of eigenvalues. Authors: Ravi Varadhan, Johns Hopkins University.

el.convex Empirical likelihood ratio tests for means. Authors: Dan Yang, Dylan Small.

elmNN Implementation of ELM (Extreme Learning Machine) algorithm for SLFN (Single Hidden Layer Feedforward Neural Networks). Author: Alberto Gosso.

emudata Datasets for the **emu** package. Authors: Jonathan Harrington, Tina John (package build) and others.

enaR Tools ecological network analysis (ena). Authors: M.K. Lau, S.R. Borrett, D.E. Hines.

epr Easy polynomial regression. Author: Emmanuel Arnhold.

evora Epigenetic Variable Outliers for Risk prediction Analysis. Author: Andrew E Teschendorff.

expoRkit Expokit in R. Authors: Roger B. Sidje, Niels Richard Hansen.

extraTrees ExtraTrees method. Author: Jaak Simm.

extrafont Tools for using fonts. Author: Winston Chang.

extrafontdb Database for the **extrafont** package. Author: Winston Chang.

ezglm Selects significant non-additive interaction between two variables using fast GLM implementation. Author: Yi Yang.

fanplot Visualisations of sequential probability distributions. Author: Guy J. Abel.

fastSOM Fast Calculation of Spillover Measures. Authors: Stefan Kloessner, Sven Wagner.

fdasrvf Elastic Functional Data Analysis. Author: J. Derek Tucker.

fdrci Permutation-based FDR Point and Confidence Interval Estimation. Author: Joshua Millstein.

finebalance Approximate fine balance when exact fine balance is not achievable. Author: Dan Yang.

fitDRC Fitting Density Ratio Classes. Authors: Simon L. Rinderknecht and Peter Reichert.

flare Family of Lasso Regression. Authors: Xingguo Li, Tuo Zhao, Lie Wang, Xiaoming Yuan and Han Liu.

flubase Baseline of mortality free of influenza epidemics. Authors: Nunes B, Natario I and Carvalho L.

fma Data sets from "Forecasting: methods and applications" by Makridakis, Wheelwright & Hyndman (1998). Author: Rob J Hyndman. In views: *Econometrics*, *TimeSeries*.

fmt Variance estimation of FMT method (Fully Moderated t-statistic). Authors: Lianbo Yu, The Ohio State University.

fontcm Computer Modern font for use with **extrafont** package. Authors: Winston Chang, Alexej Kryukov, Paul Murrell.

forensic Statistical Methods in Forensic Genetics. Author: Miriam Marusiakova.

fpow Computing the noncentrality parameter of the noncentral *F* distribution. Author: Ali Baharev.

frbs Fuzzy rule-based systems. Authors: Lala Septem Riza, Christoph Bergmeir, Francisco Herrera Triguero, and Jose Manuel Benitez.

freeknotsplines Free-Knot Splines. Authors: Steven Spiriti, Philip Smith, Pierre Lecuyer.

frontiles Partial frontier efficiency analysis. Authors: Abdelaati Daouia, Thibault Laurent.

frt Full Randomization Test. Authors: Giangiacomo Bravo, Lucia Tamburino.

fugeR FUzzy GEnetic, a machine learning algorithm to construct prediction model based on fuzzy logic. Author: Alexandre Bujard.

fwi.fbp Fire Weather Index System and Fire Behaviour Prediction System Calculations. Authors: Xianli Wang, Alan Cantin, Marc-Andre Parisien, Mike Wotton, Kerry Anderson, and Mike Flannigan.

gMWT Generalized Mann-Whitney Type Tests. Authors: Daniel Fischer, Hannu Oja.

gProfileR g:ProfileR. Authors: Juri Reimand, Raivo Kolde, Tambet Arak.

gamclass Functions and data for a course on modern regression and classification. Author: John Maindonald.

gamlss Generalized Additive Models for Location Scale and Shape. Authors: Mikis Stasinopoulos, Bob Rigby with contributions from Calliope Akantziliotou and Vlasis Voudouris. In view: *Econometrics*.

gbs Generalized Birnbaum-Saunders Distributions. Authors: Michelli Barros, Victor Leiva and Gilberto A. Paula. In view: *Distributions*.

gdimap Generalized Diffusion Magnetic Resonance Imaging. Author: Adelino Ferreira da Silva.

gearman R interface to the Gearman Job Server. Author: Jeffrey Horner.

geeM Fit Generalized Estimating Equations. Authors: Lee McDaniel and Nick Henderson.

gemtc GeMTC network meta-analysis. Authors: Gert van Valkenhoef, Joel Kuiper.

gemtc.jar GeMTC Java binary. Authors: Gert van Valkenhoef, Joel Kuiper.

genSurv Generating multi-state survival data. Authors: Artur Agostinho Araújo, Luís Meira-Machado and Susana Faria. In view: *Survival*.

geneListPie Profiling a gene list into GOslim or KEGG function pie. Author: Xutao Deng.

geneSignatureFinder A Gene-signatures finder tools. Authors: Stefano M. Pagnotta, Michele Ceccarelli.

genlasso Path algorithm for generalized lasso problems. Authors: Ryan J. Tibshirani, Taylor B. Arnold.

genomatic Manages microsatellite projects. Creates 96-well maps, genotyping submission forms, rerun management, and import into statistical software. Author: Brian J. Knaus.

geomorph Geometric morphometric analysis of 2d/3d landmark data. Authors: Dean Adams, Erik Otarola-Castillo. In view: *Phylogenetics*.

geotopbricks Analyzes raster maps as input/output files from the Hydrological Distributed Model GEOtop. Authors: Emanuele Cordano, Daniele Andreis, Fabio Zottele.

ggmcmc Graphical tools for analyzing Markov Chain Monte Carlo simulations from Bayesian inference. Author: Xavier Fernández i Marín. In view: *Bayesian*.

ggparallel Variations of Parallel Coordinate Plots for Categorical Data. Authors: Heike Hofmann, Marie Vendettuoli.

ggsubplot Explore complex data by embedding subplots within plots. Authors: Garrett Grolemund, Hadley Wickham.

globalboosttest Testing the additional predictive value of high-dimensional data. Authors: Anne-Laure Boulesteix, Torsten Hothorn. In view: *Survival*.

gnmf Generalized Non-negative Matrix Factorization. Authors: Jose M. Maisog, Guoli Wang, Karthik Devarajan.

gppois Gaussian Processes for Poisson-noised Data. Author: Charles R. Hogg III.

gtable Arrange grobs in tables. Author: Hadley Wickham.

gwerAM Controlling the genome-wide type I error rate in association mapping experiments. Authors: Benjamin Stich, Bettina Mueller, Hans-Peter Piepho.

hbsae Hierarchical Bayesian Small Area Estimation. Author: Harm Jan Boonstra. In view: *Bayesian*.

heatmapFit Heatmap Fit Statistic For Binary Dependent Variable Models. Authors: Justin Esarey and Andrew Pierce.

hierNet A Lasso for Hierarchical Interactions. Authors: Jacob Bien and Rob Tibshirani.

hierarchicalDS Functions for performing hierarchical analysis of distance sampling data. Author: P.B. Conn.

hmeasure The H-measure and other scalar classification performance metrics. Authors: Christoforos Anagnostopoulos and David J. Hand.

hmmm Hierarchical multinomial marginal models. Authors: Roberto Colombi, Sabrina Giordano, Manuela Cazzaro.

holdem Texas Holdem simulator. Author: Frederic Paik Schoenberg.

homeR Functions useful for building physics. Author: Neurobat AG.

hwriterPlus Extending the **hwriter** Package. Author: David Scott.

hypothesestest Confidence Intervals and Tests of Statistical Hypotheses. Authors: Chengfeng Liu, Huiqing Liu, Yingyan Liang, Ruibin Feng.

hzar Hybrid Zone Analysis using R. Author: Graham Derryberry.

iBUGS An Interface to R2WinBUGS/R2jags by gWidgets. Authors: Yihui Xie and Jiebiao Wang.

icensmis Study Design and Data Analysis in the presence of error-prone diagnostic tests and self-reported outcomes. Authors: Xiangdong Gu and Raji Balasubramanian.

icomp ICOMP criterion. Author: Jake Ferguson.

igraphtosonia Convert iGraph graps to SoNIA '.son' files. Author: Sean J Westwood.

infutil Information Utility. Author: Kristian E. Markon.

insol Solar Radiation. Author: Javier G. Corripio.

intsvy Data Manager of International Assessment Studies of Student Performance. Author: Daniel Caro.

isopat Calculation of isotopic pattern for a given molecular formula. Author: Martin Loos.

kSamples K-Sample Rank Tests and their Combinations. Authors: Fritz Scholz and Angie Zhu.

kelvin Calculate solutions to Kelvin differential equation using Kelvin functions. Author: Andrew J Barbour.

kitagawa Model the spectral response of a closed water-well to harmonic strains at seismic frequencies. Author: Andrew J Barbour.

klin Linear equations with Kronecker structure. Author: Tamas K Papp.

knitcitations Citations for knitr markdown files. Author: Carl Boettiger.

kobe Tools for providing advice for the Tuna Regional Fisheries Management Organisations. Author: Laurence Kell.

labeling Axis Labeling. Author: Justin Talbot.

lambda.r Functional programming in R. Author: Brian Lee Yung Rowe.

lavaan Latent Variable Analysis. Authors: Yves Rosseel [aut, cre], Daniel Oberski [ctb], Jarrett Byrnes [ctb], Leonard Vanbrabant [ctb], Victoria Savalei [ctb], Ed Merkle [ctb], Michael Hallquist [ctb], Mijke Rhemtulla [ctb], Myrsini Katsikatsou [ctb]. In view: *Psychometrics*.

lavaan.survey Complex survey structural equation modeling (SEM). Author: Daniel Oberski.

ldlasso LD LASSO Regression for SNP Association Study. Author: Samuel G. Younkin.

ldr Methods for likelihood-based dimension reduction in regression. Authors: Kofi Placid Adragni, Andrew Raim.

linLIR linear Likelihood-based Imprecise Regression. Author: Andrea Wiencierz.

lineup Lining up two sets of measurements. Author: Karl W Broman.

lint Tools to check R code style. Author: Andrew Redd.

lmec Linear Mixed-Effects Models with Censored Responses. Authors: Florin Vaida and Lin Liu. In view: *Survival*.

logmult Log-multiplicative models, including association models. Author: Milan Bouchet-Valat.

logregperm Inference in Logistic Regression. Author: Douglas M. Potter.

loop loop decomposition of weighted directed graphs for life cycle analysis, providing flexbile network plotting methods, and analyzing food chain properties in ecology. Author: Youhua Chen.

lpint Local polynomial esitmators of intensity function or its derivatives. Author: Feng Chen.

lsmeans Least-squares means. Author: Russell V. Lenth.

mRMRe Parallelized mRMR ensemble feature selection. Authors: Nicolas De Jay, Simon Papillon-Cavanagh, Benjamin Haibe-Kains.

maRketSim Market simulator for R. Author: Ari Friedman. In view: *Finance*.

magicaxis Pretty scientific plotting with minor-tick and log minor-tick support. Author: Aaron Robotham.

mapplots Data visualisation on maps. Author: Hans Gerritsen.

maxLinear Conditional Samplings for Max-Linear Models. Author: Yizao Wang.

mcclust Process an MCMC Sample of Clusterings. Author: Arno Fritsch. In view: *Cluster*.

mcll Monte Carlo Local Likelihood Estimation. Authors: Minjeong Jeon, Cari Kaufman, and Sophia Rabe-Hesketh.

mcpd Tools to analyse and use passport data for biological collections. Author: Reinhard Simon.

mcr Method Comparison Regression. Authors: Ekaterina Manuilova, Andre Schuetzenmeister, Fabian Model.

mded Measuring the difference between two empirical distributions. Author: Hideo Aizaki.

mederrRank Bayesian Methods for Identifying the Most Harmful Medication Errors. Authors: Sergio Venturini, Jessica Myers.

metRology Support for metrological applications. Author: Stephen L R Ellison.

metamisc Diagnostic and prognostic meta analysis (metamisc). Author: Thomas Debray.

miRada MicroRNA Microarray Data Analysis. Author: Bin Wang.

migest Useful R code for the Estimation of Migration. Author: Guy J. Abel.

minerva Maximal Information-Based Nonparametric Exploration R package for Variable Analysis. Authors: Michele Filosi [aut, cre], Roberto Visintainer [aut], Davide Albanese [aut], Samantha Riccadonna [ctb], Giuseppe Jurman [ctb], Cesare Furlanello [ctb].

minxent Entropy Optimization Distributions. Author: Senay Asma.

mixfdr Computes false discovery rates and effect sizes using normal mixtures. Authors: Omkar Muralidharan, with many suggestions from Bradley Efron.

mlPhaser Multi-Locus Haplotype Phasing. Author: Dave T. Gerrard.

mlica2 Independent Component Analysis using Maximum Likelihood. Author: Andrew Teschendorff.

mmeln Estimation of multinormal mixture distribution. Author: Charles-Edouard Giguere.

mmeta Multivariate Meta-Analysis Using Sarmanov Beta Prior Distributions. Authors: Sheng Luo, Yong Chen, Haitao Chu, Xiao Su.

mmm Multivariate Marginal Models. Authors: Ozgur Asar, Ozlem Ilk.

motmot Models of Trait Macroevolution on Trees. Authors: Gavin Thomas, Rob Freckleton. In view: *Phylogenetics*.

mpa CoWords Method. Author: Daniel Hernando Rodriguez and Campo Elias Pardo.

msap Statistical analysis for Methylation-sensitive Amplification Polymorphism data. Author: Andres Perez-Figueroa.

mtcreator Creating MAGE-TAB files using mtcreator. Author: Fabian Grandke.

mtsdi Multivariate time series data imputation. Authors: Washington Junger and Antonio Ponce de Leon.

multgee GEE Solver for Correlated Nominal or Ordinal Multinomial Responses. Author: Anestis Touloumis.

multibiplotGUI Multibiplot Analysis in R. Authors: Ana Belen Nieto Librero, Nora Baccala, Purificacion Vicente Galindo, Purificacion Galindo Villardon.

multisensi Multivariate Sensitivity Analysis. Authors: Matieyendou Lamboni, Herve Monod.

muscle Multiple Sequence Alignment. Author: Algorithm by Robert C. Edgar. R port by Alex T. Kalinka.

mvShapiroTest Generalized Shapiro-Wilk test for multivariate normality. Authors: Elizabeth Gonzalez Estrada, Jose A. Villasenor Alva.

mvc Multi-View Clustering. Author: Andreas Maunz.

mvsf Shapiro-Francia Multivariate Normality Test. Author: David Delmail.

mvtmeta Multivariate meta-analysis. Author: Han Chen.

namespace Provide namespace managment functions not (yet) present in base R. Authors: Winston Chang, Daniel Adler, Hadley Wickham, Gregory R. Warnes, R Core Team.

ncg Computes the noncentral gamma function. Authors: Daniel Furtado Ferreira, Izabela Regina Cardoso de Oliveira and Fernando Henrique Toledo.

ngspatial Classes for Spatial Data. Author: John Hughes.

nlADG Regression in the Normal Linear ADG Model. Authors: Gruber, Lutz F.

nlmrt Functions for nonlinear least squares solutions. Author: John C. Nash.

nlts (non)linear time series analysis. Author: Ottar N. Bjornstad.

nontarget Detecting, combining and filtering isotope, adduct and homologue series relations in high-resolution mass spectrometry (HRMS) data. Author: Martin Loos.

nopp Nash Optimal Party Positions. Authors: Luigi Curini, Stefano M. Iacus.

normwhn.test Normality and White Noise Testing. Author: Peter Wickham.

notifyR Send push notifications to your smartphone via pushover.net. Author: Torben Engelmeyer.

npmv Nonparametric Comparison of Multivariate Samples. Author: Woodrow Burchett.

numbers Number-theoretic Functions. Author: Hans W Borchers.

obliclus Cluster-based factor rotation. Author: Michio Yamamoto.

ocomposition Gibbs sampler for ordered compositional data. Authors: Arturas Rozenas, Duke University.

oem Orthogonalizing Expectation maximization. Author: Bin Dai.

omd Filter the molecular descriptors for QSAR. Author: Bin Ma.

oncomodel Maximum likelihood tree models for oncogenesis. Authors: Anja von Heydebreck, contributions from Christiane Heiss.

opefimor Option Pricing and Estimation of Financial Models in R. Author: Stefano Maria Iacus. In view: *Finance*.

opmdata Example data for analysing OmniLog(R) Phenotype Microarray data. Authors: Markus Goeker, with contributions by Lea A.I. Vaas and Johannes Sikorski.

orcutt Estimate procedure in case of first order autocorrelation. Authors: Stefano Spada, Matteo Quartagno, Marco Tamburini.

oro.pet Rigorous — Positron Emission Tomography. Author: Brandon Whitcher.

p2distance Welfare's Synthetic Indicator. Authors: A.J. Perez-Luque; R. Moreno; R. Perez-Perez and F.J. Bonet.

pGLS Generalized Least Square in comparative Phylogenetics. Authors: Xianyun Mao and Timothy Ryan.

pa Performance Attribution for Equity Portfolios. Authors: Yang Lu and David Kane. In view: *Finance*.

pacbpred PAC-Bayesian Estimation and Prediction in Sparse Additive Models. Author: Benjamin Guedj. In view: *Bayesian*.

pander An R pandoc writer. Author: Gergely Daróczi.

parspatstat Parallel spatial statistics. Author: Jonathan Lee.

partitionMetric Compute a distance metric between two partitions of a set. Authors: David Weisman, Dan Simovici.

parviol Author: Jaroslav Myslivec.

pbdBASE Programming with Big Data — Core pbd Classes and Methods. Authors: Drew Schmidt, Wei-Chen Chen, George Ostrouchov, Pragneshkumar Patel. In view: *HighPerformanceComputing*.

pbdDMAT Programming with Big Data — Distributed Matrix Computation. Authors: Drew Schmidt, Wei-Chen Chen, George Ostrouchov, Pragneshkumar Patel. In view: *HighPerformanceComputing*.

pbdMPI Programming with Big Data — Interface to MPI. Authors: Wei-Chen Chen, George Ostrouchov, Drew Schmidt, Pragneshkumar Patel, Hao Yu. In view: *HighPerformanceComputing*.

pbdSLAP Programming with Big Data — Scalable Linear Algebra Packages. Authors: Wei-Chen Chen [aut, cre], Drew Schmidt [aut], George Ostrouchov [aut], Pragneshkumar Patel [aut], ScaLAPACK [ctb]. In view: *HighPerformanceComputing*.

pbivnorm Vectorized Bivariate Normal CDF. Authors: Fortran code by Alan Genz. R code by Brenton Kenkel, based on Adelchi Azzalini's **mnormt** package.

pcenum Permutations and Combinations Enumeration. Author: Benjamin Auder.

penDvine Flexible Pair-Copula Estimation in D-vines with Penalized Splines. Author: Christian Schellhase.

peplib Peptide Library Analysis Methods. Author: Andrew White.

perry Resampling-based prediction error estimation for regression models. Author: Andreas Alfons.

pesticides Analysis of single serving and composite pesticide residue measurements. Author: David M Diez.

pheno2geno Generating genetic markers and maps from molecular phenotypes. Authors: Konrad Zych and Danny Arends.

phonR R tools for phoneticians and phonologists. Author: Daniel R. McCloy.

phonTools Functions for phonetics in R. Author: Santiago Barreda.

pkgutils Utilities for creating R packages. Author: Markus Goeker.

planor Generation of regular factorial designs. Authors: Hervé Monod, Annie Bouvier, André Kobilinsky.

plmm Partially Linear Mixed Effects Model. Author: Ohinata Ren.

pln Polytomous logit-normit (graded logistic) model estimation. Authors: Carl F. Falk and Harry Joe.

plotKML Visualization of spatial and spatio-temporal objects in Google Earth. Authors: Tomislav Hengl, Contributions by: Pierre Roudier, Dylan Beaudette, Daniel Nuest. In view: *Spatial*.

plotSEMM Graphing nonlinear latent variable interactions in SEMM. Authors: Bethany E. Kok, Jolynn Pek, Sonya Sterba and Dan Bauer.

plsdepot Partial Least Squares (PLS) Data Analysis Methods. Author: Gaston Sanchez.

pnmtrem Probit-Normal Marginalized Transition Random Effects Models. Authors: Ozgur Asar, Ozlem Ilk.

poibin The Poisson Binomial Distribution. Author: Yili Hong. In view: *Distributions*.

poisson.glm.mix Fit high dimensional mixtures of Poisson GLM's. Authors: Panagiotis Papastamoulis, Marie-Laure Martin-Magniette, Cathy Maugis-Rabusseau.

poistweedie Poisson-Tweedie exponential family models. Authors: David Pechel Cactcha, Laure Pauline Fotso and Celestin C Kokonendji. In view: *Distributions*.

popReconstruct Reconstruct population counts, fertility, mortality and migration rates of human populations of the recent past. Author: Mark C. Wheldon.

postgwas GWAS Post-Processing Utilities. Author: Milan Hiersche.

powerGWASinteraction Power Calculations for Interactions for GWAS. Author: Charles Kooperberg.

ppMeasures Point pattern distances and prototypes. Authors: David M Diez, Katherine E Tranbarger Freier, and Frederic P Schoenberg.

ppcor Partial and Semi-partial (Part) correlation. Author: Seongho Kim.

pragma Provides a pragma / directive / keyword syntax for R. Author: Christopher Brown.

prettyGraphs Publication-quality graphics. Author: Derek Beaton.

prevR Estimating regional trends of a prevalence from a DHS. Authors: Joseph Larmarange — CEPED (Universite Paris Descartes Ined IRD) IRD, with fundings from ANRS and IRD, and technical support from LYSIS (info@lysis-consultants.fr).

profanal Implements profile analysis described in Davison & Davenport (2002). Author: Christopher David Desjardins.

protViz Visualizing and Analyzing Mass Spectrometry Related Data in Proteomics. Authors: Christian Panse, Jonas Grossmann.

protiq Protein (identification and) quantification based on peptide evidence. Authors: Sarah Gerster and Peter Buehlmann.

protr Protein Sequence Feature Extraction with R. Authors: Xiao Nan, Dongsheng Cao, Qingsong Xu, Yizeng Liang.

psytabs Produce well-formatted tables for psychological research. Authors: Johannes Beller, Soeren Kliem.

pvar p-variation. Author: Vygantas Butkus.

quadrupen Sparsity by Worst-Case Quadratic Penalties. Author: Julien Chiquet.

r2stl Visualizing data using a 3D printer. Authors: Ian Walker and José Gama.

rAltmetric Retrieves altmerics data for any published paper from altmetrics.com. Author: Karthik Ram.

rHpcc Interface between HPCC and R. Author: Dinesh Shetye.

rImpactStory Retrieves altmetrics from ImpactStory. See `http://impactstory.org` for more about the metrics. Authors: Karthik Ram, Scott Chamberlain.

rJython R interface to Python via Jython. Authors: G. Grothendieck and Carlos J. Gil Bellosta (authors of Jython itself are Jim Hugunin, Barry Warsaw, Samuele Pedroni, Brian Zimmer, Frank Wierzbicki and others; Bob Ippolito is the author of the simplejson Python module).

rPlant R interface to the iPlant Discovery Environment. Authors: Barb Banbury, Kurt Michels, Jeremy M. Beaulieu, Brian O'Meara.

randomForestSRC Random Forests for Survival, Regression, and Classification (RF-SRC). Authors: Hemant Ishwaran, Udaya B. Kogalur.

randomizeBE Function to create a random list for crossover studies. Author: D. Labes.

rbundler Manage an application's dependencies systematically and repeatedly. Author: Yoni Ben-Meshulam.

rcqp Interface to the Corpus Query Protocol. Authors: Bernard Desgraupes, Sylvain Loiseau.

rdd Regression Discontinuity Estimation. Author: Drew Dimmery.

rdetools Relevant Dimension Estimation (RDE) in Feature Spaces. Author: Jan Saputra Mueller. In view: *MachineLearning*.

rdyncall Improved Foreign Function Interface (FFI) and Dynamic Bindings to C Libraries (e.g. OpenGL). Author: Daniel Adler.

reams Resampling-Based Adaptive Model Selection. Authors: Philip Reiss and Lei Huang.

rebird Interface to eBird. Authors: Rafael Maia, Scott Chamberlain.

reccsim Simulation of Rare Events Case-Control Studies. Author: Christian Westphal.

regsubseq Detect and Test Regular Sequences and Subsequences. Author: Yanming Di.

rentrez Entrez in R. Author: David Winter.

repolr Repeated measures proportional odds logistic regression. Author: Nick Parsons.

restlos Robust estimation of location and scatter. Authors: Steffen Liebscher and Thomas Kirschstein.

retimes Reaction Time Analysis. Author: Davide Massidda.

review Manage Review Logs. Author: Tim Bergsma.

rfigshare An R interface to figshare.com. Authors: Carl Boettiger, Scott Chamberlain, Karthik Ram, Edmund Hart.

rgexf Build GEXF network files. Authors: George Vega Yon, Jorge Fabrega Lacoa.

ridge Ridge Regression with automatic selection of the penalty parameter. Author: Erika Cule.

ritis Taxonomic search of ITIS data. Author: Scott Chamberlain.

rkt Mann-Kendall test, Seasonal and Regional Kendall Tests. Author: Aldo Marchetto.

rlandscape Generates random landscapes with specifiable spatial characteristics. Authors: Gregor Passolt, Miranda J. Fix, Sandor F. Toth.

rmmseg4j R interface to the Java Chinese word segmentation system of mmseg4j. Author: Huang Ronggui.

rngtools Utility functions for working with Random Number Generators. Author: Renaud Gaujoux.

robustgam Robust Estimation for Generalized Additive Models. Author: Raymond K. W. Wong.

robustloggamma Robust estimation of the generalized log gamma model. Authors: Claudio Agostinelli, Alfio Marazzi, V.J. Victor and Alex Randriamiharisoa.

robustreg Robust Regression Functions. Author: Ian M. Johnson.

ropensnp Interface to OpenSNP API methods. Author: Scott Chamberlain.

rpf Response Probability Functions. Authors: Joshua Pritikin [cre, aut], Jonathan Weeks [ctb]. In view: *Psychometrics*.

rrcovHD Robust multivariate Methods for High Dimensional Data. Author: Valentin Todorov.

rriskBayes Predefined Bayes models fitted with Markov chain Monte Carlo (MCMC) (related to the 'rrisk' project). Authors: Natalia Belgorodski, Matthias Greiner, Alexander Engelhardt.

rseedcalc Estimating the proportion of genetically modified stacked seeds in seedlots via multinomial group testing. Authors: Kevin Wright, Jean-Louis Laffont.

rspa Adapt numerical records to fit (in)equality restrictions with the Successive Projection Algorithm. Author: Mark van der Loo.

rts Raster time series analysis. Author: Babak Naimi.

rvertnet Search VertNet database from R. Authors: Scott Chamberlain, Vijay Barve.

rworldxtra Country boundaries at high resolution. Author: Andy South.

sExtinct Calculates the historic date of extinction given a series of sighting events. Author: Christopher Clements.

samplingVarEst Sampling Variance Estimation. Authors: Emilio Lopez Escobar, Ernesto Barrios Zamudio.

sddpack Semidiscrete Decomposition. Authors: Tamara G. Kolda, Dianne P. O'Leary.

sdnet Soft Discretization-based Bayesian Network Inference. Author: Nikolay Balov.

seem Simulation of Ecological and Environmental Models. Author: Miguel F. Acevedo.

selectr Translate CSS Selectors to XPath Expressions. Authors: Simon Potter, Simon Sapin, Ian Bicking.

separationplot Separation Plots. Authors: Brian D. Greenhill, Michael D. Ward and Audrey Sacks.

seq2R Simple method to detect compositional changes in genomic sequences. Authors: Nora M. Villanueva, Marta Sestelo and Javier Roca-Pardinas.

sig Print function signatures. Author: Richard Cotton.

sigclust Statistical Significance of Clustering. Authors: Hanwen Huang, Yufeng Liu and J. S. Marron. In view: *Cluster*.

sigora SIGnature OverRepresentation Analysis. Authors: Amir B.K. Foroushani, Fiona S.L. Brinkman, David J. Lynn.

sirad Functions for calculating daily solar radiation and evapotranspiration. Author: Jedrzej S. Bojanowski.

sisus Stable Isotope Sourcing using Sampling. Author: Erik Barry Erhardt.

skewt The Skewed Student-t Distribution. Authors: Robert King, with contributions from Emily Anderson. In view: *Distributions*.

smart Sparse Multivariate Analysis via Rank Transformation. Authors: Fang Han, Han Liu.

smco A simple Monte Carlo optimizer using adaptive coordinate sampling. Author: Juan David Velasquez.

sme Smoothing-splines Mixed-effects Models. Author: Maurice Berk.

smirnov Provides two taxonomic coefficients from E. S. Smirnov "Taxonomic analysis" (1969) book. Author: Alexey Shipunov (with help of Eugenij Altshuler).

sms Spatial Microsimulation. Author: Dimitris Kavroudakis.

snort Social Network-Analysis On Relational Tables. Authors: Eugene Dubossarsky and Mark Norrie.

soilwater Implements parametric formulas for soil water retention or conductivity curve. Author: Emanuele Cordano.

spTimer Spatio-Temporal Bayesian Modelling Using R. Authors: K. Shuvo Bakar and Sujit K. Sahu. In view: *Bayesian*.

spa Implements The Sequential Predictions Algorithm. Author: Mark Culp.

sparseHessianFD Interface to ACM TOMS Algorithm 636, for computing sparse Hessians. Authors: R interface code by Michael Braun; original Fortran code by Thomas F. Coleman, Burton S. Garbow and Jorge J. More.

sperrorest Spatial Error Estimation and Variable Importance. Author: Alexander Brenning.

sphereplot Spherical plotting. Author: Aaron Robotham.

spsmooth An Extension Package for **mgcv**. Authors: Wesley Burr, with contributions from Karim Rahim.

squash Color-based plots for multivariate visualization. Author: Aron Eklund.

sra Selection Response Analysis. Author: Arnaud Le Rouzic.

stargazer LaTeX code for well-formatted regression and summary statistics tables. Author: Marek Hlavac.

stepp Subpopulation Treatment Effect Pattern Plot (STEPP). Authors: Wai-ki Yip, with contributions from Ann Lazar, David Zahrieh, Chip Cole, Ann Lazar, Marco Bonetti, and Richard Gelber.

stocc Fit a spatial occupancy model via Gibbs sampling. Author: Devin S. Johnson.

stppResid Perform residual analysis on space-time point process models. Author: Robert Clements.

sudokuplus Sudoku Puzzle ($9*9$, $12*12$, $16*16$) Generator and Solver. Authors: Zhengmairuo Gan, Yuzhen Hua, Maosheng Zhang, Caiyan Lai.

survIDINRI IDI and NRI for comparing competing risk prediction models with censored survival data. Authors: Hajime Uno, Tianxi Cai. In view: *Survival*.

survivalROC Time-dependent ROC curve estimation from censored survival data. Authors: Patrick J. Heagerty, packaging by Paramita Saha. In view: *Survival*.

svapls Surrogate variable analysis using partial least squares in a gene expression study. Authors: Sutirtha Chakraborty, Somnath Datta and Susmita Datta.

sybilSBML SBML Integration in Package **sybil**. Author: Gabriel Gelius-Dietrich.

symbols Symbol plots. Author: Jaroslav Myslivec.

taRifx.geo Collection of various spatial functions. Author: Ari B. Friedman.

tabplotd3 Interactive inspection of large data. Authors: Edwin de Jonge and Martijn Tennekes.

teigen Model-based clustering and classification with the multivariate t-distribution. Authors: Jeffrey L. Andrews, Paul D. McNicholas. In view: *Cluster*.

texreg Conversion of R regression output to LaTeX tables. Author: Philip Leifeld.

tiff Read and write TIFF images. Author: Simon Urbanek.

tightClust Tight Clustering. Authors: George C. Tseng, Wing H. Wong.

tilting Variable selection via Tilted Correlation Screening algorithm. Author: Haeran Cho.

timeROC Time-dependent ROC curve and AUC for censored survival data. Author: Paul Blanche.

tmg Truncated Multivariate Gaussian Sampling. Author: Ari Pakman.

tmle Targeted Maximum Likelihood Estimation. Authors: Susan Gruber, in collaboration with Mark van der Laan.

transmission Continuous time infectious disease models on individual data. Authors: Alun Thomas, Andrew Redd.

transnet Conducts transmission modeling on a bayesian network. Author: Alun Thomas.

trex Truncated exact test for two-stage case-control design for studying rare genetic variants. Authors: Schaid DJ, Sinnwell JP.

trifield Some basic facilities for ternary fields and plots. Author: Tim Keitt.

trustOptim Trust region nonlinear optimization, efficient for sparse Hessians. Author: Michael Braun. In view: *Optimization*.

tslars Least angle regression for time series analysis. Author: Sarah Gelper. In view: *TimeSeries*.

two.stage.boot Two-stage cluster sample bootstrap algorithm. Author: Patrick Zimmerman.

twoStageGwasPower Compute thresholds and power for two-stage gwas. Author: Dirk F Moore.

upclass Updated Classification Methods using Unlabelled Data. Authors: Niamh Russell, Laura Cribbin, Thomas Brendan Murphy.

usdm Uncertainty analysis for species distribution models. Author: Babak Naimi.

utility Construct, Evaluate and Plot Value and Utility Functions. Author: Peter Reichert with contributions by Nele Schuwirth.

validator External and Internal Validation Indices. Author: Marcus Scherl.

vcf2geno Efficiently Read Variant Call Format (VCF) into R. Authors: Xiaowei Zhan and Dajiang Liu, with contributions of Jean-loup Gailly, Mark Adler, Julian Seward and Heng Li.

vegclust Fuzzy clustering of vegetation data. Author: Miquel De Caceres.

violinmplot Combination of violin plot with mean and standard deviation. Author: Raphael W. Majeed.

vmv Visualization of Missing Values. Author: Waqas Ahmed Malik.

vows Voxelwise semiparametrics. Authors: Philip Reiss, Yin-Hsiu Chen, Lei Huang, and Lan Huo.

vwr Useful functions for visual word recognition research. Author: Emmanuel Keuleers.

wSVM Weighted SVM with boosting algorithm for improving accuracy. Authors: SungHwan Kim and Soo-Heang Eo.

waterData Retrieval, Analysis, and Anomaly Calculation of Daily Hydrologic Time Series Data. Authors: Karen R. Ryberg and Aldo V. Vecchia.

wavemulcor Wavelet routine for multiple correlation. Author: Javier Fernandez-Macho.

weathermetrics Functions to convert between weather metrics. Authors: Brooke Anderson and Roger Peng.

wgsea Wilcoxon based gene set enrichment analysis. Author: Chris Wallace.

widals Weighting by Inverse Distance with Adaptive Least Squares for Massive Space-Time Data. Author: Dave Zes.

x12GUI X12 — Graphical User Interface. Authors: Daniel Schopfhauser, Alexander Kowarik, Angelika Meraner. In view: *TimeSeries*.

xoi Tools for analyzing crossover interference. Authors: Karl W Broman, Il youp Kwak.

zendeskR Zendesk API Wrapper. Author: Tanya Cashorali.

zyp Zhang + Yue-Pilon trends. Authors: David Bronaugh, Arelia Werner for the Pacific Climate Impacts Consortium.

Other changes

- The following packages were moved to the Archive: **AIGIS**, **Biograph**, **BradleyTerry**, **CCMtools**, **CGene**, **CONOR**, **COZIGAM**, **CompetingRiskFrailty**, **Covpath**, **DCluster**, **DDHFm**, **DOSim**, **DeducerMMR**, **Depela**, **DescribeDisplay**, **Devore5**, **EMT**, **ElectroGraph**, **EuclideanMaps**, **FourierDescriptors**, **GWAtoolbox**, **Geneclust**, **HFWutils**, **IFP**, **IQMNMR**, **MCE**, **MCLIME**, **MFDA**, **NMF**, **OptionsPdf**, **PairedData**, **RFA**, **RLadyBug**, **RScaLAPACK**, **RSiteSearch**, **Rassoc**, **Ratings**, **RcmdrPlugin.EHESsampling**, **RcmdrPlugin.SensoMineR**, **RcmdrPlugin.SurvivalT**, **Rfun**, **Rpad**, **SNPMaP**, **SNPMaP.cdm**, **SQLiteMap**, **SSSR**, **SV**, **SeqKnn**, **SpatialEpi**, **ThreeGroups**, **TwoWaySurvival**, **UScensus2000**, **UScensus2000add**, **VhayuR**, **accuracy**, **afc**, **agilp**, **amba**, **amer**, **anm**, **backfitRichards**, **belief**, **biGraph**, **brainwaver**, **bspec**, **bwsurvival**, **chplot**, **clac**, **clustTool**,

clusterfly, **clustvarsel**, **codep**, **colorout**, **compOverlapCorr**, **concord**, **copulaedas**, **covRobust**, **crossdes**, **crosshybDetector**, **cthresh**, **cwhmisc**, **cyphid**, **dcens**, **diamonds**, **digitize**, **dtt**, **dyad**, **edci**, **envelope**, **epinet**, **exactmaxsel**, **extfunnel**, **favir**, **ffmanova**, **financial**, **fptdApprox**, **frbf**, **fuzzyOP**, **gRC**, **gafit**, **gbev**, **gcmrec**, **geneARMA**, **hot**, **ipptoolbox**, **iteRates**, **iv**, **jit**, **knn**, **knncat**, **labeltodendro**, **latticedl**, **lda.cv**, **lodplot**, **mapLD**, **maptree**, **margLikArrogance**, **mblm**, **mecdf**, **mimR**, **moc**, **mosaicManip**, **mrp**, **mrpdata**, **mrt**, **msBreast**, **msDilution**, **msProcess**, **msProstate**, **mtsc**, **networksis**, **nfda**, **nonbinROC**, **nutshellDE**, **onemap**, **paltran**, **panel**, **papply**, **partitionMap**, **pcaPA**, **permax**, **pgfSweave**, **phpSerialize**, **phybase**, **qAnalyst**, **quaternions**, **rTOFsPRO**, **rWMBAT**, **rake**, **richards**, **rrp**, **rrv**, **rsdepth**, **rtv**, **satin**, **scaleCoef**, **sdtalt**, **seas**, **simco**, **similarityRichards**, **skellam**, **skills**, **soil.spec**, **somplot**, **spssDDI**, **stream.net**, **surveillance**, **swst**, **tikzDevice**, **triads**, **truncgof**, **twslm**, **uncompress**, **voronoi**, **xterm256**

- The following packages were resurrected from the Archive: **MAMA**, **NBPSeq**, **Peak2Trough**, **RPPanalyzer**, **SAPP**, **Sim.DiffProcGUI**, **SimHap**, **TSTutorial**, **copuladaes**, **dummies**, **geoPlot**, **gmvalid**, **latentnet**, **lcd**, **mapReduce**, **miniGUI**, **muStat**, **nonrandom**, **pheno**, **statnet**, **treelet**, **ttime**.

- The following packages had to be removed: **Deducer**, **PBSmapping**, **fEcofin**, **ggdendro**.

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org

News from the Bioconductor Project

Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center

Bioconductor 2.11 was released on 3 October 2012. It is compatible with R 2.15.2, and consists of 610 software packages and more than 650 up-to-date annotation packages. The release includes 58 new software packages, and enhancements to many others. Descriptions of new packages and updated NEWS files provided by current package maintainers are at http://bioconductor.org/news/bioc_2_11_release/. Start using Bioconductor and R version 2.15.2 with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

Upgrade all packages to the current release with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("BiocUpgrade")
```

Install additional packages, e.g., **VariantTools**, with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("VariantTools")
```

Explore Bioconductor at http://bioconductor.org. All packages are grouped by 'BiocViews' to identify coherent groups of packages. Each package has an html page with the descriptions and links to vignettes, reference manuals, and use statistics.

A Bioconductor Amazon Machine Instance is available; see http://bioconductor.org/help/bioconductor-cloud-ami.

Core Annotation and Software Packages

Our large collection of microarray- and organism-specific annotation packages have been updated to include current information. This release also includes the **OrganismDbi** package to integrate separate annotation resources. For example, the **Homo.sapiens** package greatly simplifies access to transcript, gene, and GO (gene ontology) annotations.

GenomicRanges and related packages, e.g., **VariantAnnotation**, **IRanges**, **Biostrings**, **Rsamtools**, **GenomicFeatures** provide an extensive, mature and extensible framework for interacting with high throughput sequence data, either as a user or package developer. Many contributed packages rely on this infrastructure for interoperable, re-usable analysis.

MotifDb, part of a new emphasis on gene regulation, offers a comprehensive annotated collection of DNA-binding motifs from popular public sources.

Other activities

Bioconductor's Annual Meeting was in Seattle, 23-25 July 2012; our European developer community meets 13-14 December in Zurich. We look forward to our next Annual Meeting on 17-19 July 2013, and to additional training and community activities advertised at http://bioconductor.org/help/events/. The active Bioconductor mailing lists (http://bioconductor.org/help/mailing-list/) 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.

R Foundation News

by Kurt Hornik

Donations and new members

Donations

Jonathan M. Lees, USA

New benefactors

Dsquare, Germany

Cybaea Limited, U.K.

New supporting members

Pavel Motuzenko, Czech Republic
Ludwig Hothorn, Germany
Gergely Darocz, Hungary

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org