R News

The Newsletter of the R Project Volume 6/4, October 2006

Editorial

by Paul Murrell

Welcome to the second regular issue of R News for 2006, which follows the release of R version 2.4.0. This issue reflects the continuing growth in R's sphere of influence, with articles covering a wide range of extensions to, connections with, and applications of R. Another sign of R's growing popularity was the success and sheer magnitude of the second useR! conference (in Vienna in June), which included over 180 presentations. Balasubramanian Narasimhan provides a report on useR! 2006 on page 45 and two of the articles in this issue follow on from presentations at useR!.

We begin with an article by Max Kuhn on the **odfWeave** package, which provides literate statistical analysis *à la* **Sweave**, but with ODF documents as the medium, rather than TEX documents. Next up is Jim Lemon with an introduction to his **plotrix** package for producing a variety of plot customisations. From graphical output to graphical user interfaces, Adrian Bowman, Crawford, and Bowman describe the **rpanel** package (introduced at useR!) which provides a friendlier wrapper on the **tcltk** package, with some nice examples of simple interactive graphics.

Matthew Pocernich takes a moment to stop and smell the CO_2 levels and describes how R is involved in some of the reproducible research that informs the climate change debate.

The next article, by Roger Peng, introduces the **filehash** package, which provides a new approach to the problem of working with large data sets in R. Robin Hankin then describes the **gsl** package, which implements an interface to some exotic mathematical functions in the GNU Scientific Library.

The final three main articles have more statistical content. Wolfgang Lederer and Helmut Küchenhoff describe the **simex** package for taking measurement error into account. Roger Koenker (another useR! presenter) discusses some non-traditional link functions for generalised linear models. And Víctor Leiva and co-authors introduce the **bs** package, which implements the Birnbaum-Saunders Distribution.

In other contributions, Susan Holmes provides a book review of Fionn Murtagh's book "Correspondence Analysis and Data Coding with Java and R" and Uwe Ligges provides an R Help Desk article on how to find the source code behind an R function.

The issue ends with our regular updates on changes to R itself, new packages on CRAN, new

Contents of this issue:

members and contributors to the R Foundation, and recent news from the Bioconductor project.

Finally, I would like to remind everyone that the next DSC conference is taking place in Auckland, New Zealand, next February 15 & 16. I hope to see you here!

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

Sweave and the Open Document Format – The odfWeave Package

by Max Kuhn

Introduction

When documenting the results of a data analysis, creating the report can easily take more time than the analysis. There are at least two approaches to this

- Create and format tables and plots, then manually add text around these elements. In this case, the analysis code has no connection to the report.
- Take a combined approach, where the analysis results and the report are created at the same time using the same source code.

In the latter case, the `Sweave` function (Leisch, 2002) is a powerful component of R that can be used to combine R code with LaTeX so that the output is embedded in the processed document.

The user could write a document in LaTeX that contains `Sweave` tags. These tags encapsulate R commands. For example, an in–line `Sweave` tag might look like `\Sexpr{sqrt(2)}`. When the LaTeX document is processed by R's `Sweave` function, the `\Sexpr` tag is a signal to R to insert the results of the command `sqrt(2)` in place of the tag.

When "weaving" with LaTeX, the user has the ability to embed textual R output, as well as more complex types of R output (such as tables). After weaving, standard LaTeX tools can be used to generate a final document file, usually in postscript or PDF format.

For example, R package vignettes are created using LaTeX and `Sweave` (Leisch, 2003). For those readers who are new to `Sweave`, typing `vignette()` at the R prompt will display a list of installed packages that contain manuals that were written using LaTeX and `Sweave`. The package sources contain the underlying `Sweave`/LaTeX files and are a good resource for getting started with `Sweave` and LaTeX.

The capabilities of `Sweave` were later extended to HTML format in the **R2HTML** package. This provided the same functionality, but uses the HTML markup language.

While `Sweave` is highly effective at producing beautiful documents that show the many features of R, there are at least two limitations created by using LaTeX or HTML as the markup language. First, it requires the user to have a moderate knowledge of the markup languages to create and produce the documents.

Secondly, using these particular markup languages limits the main file formats to Postscript, PDF or HTML. If the user is collaborating with scientists or researchers who are not comfortable with editing LaTeX or HTML, it is difficult for them to add their subject-specific insight to the document. For example, at Pfizer, R is used as a computational engine for gene expression and computational chemistry data analysis platforms. Scientists can upload their data, specify models and generate results. We would like to provide them with the results of the analysis in a format that enables them to frame the statistical output in the context of their problem

The Open Document Format and The `odfWeave` Package

The Open Document Format (ODF) is an XML–based markup language. ODF is an open, non–proprietary format that encompasses text documents, presentations and spreadsheets. Version 1.0 of the specification was finalized in May of 2005 (OASIS, 2005). One year later, the format was approved for release as an ISO and IEC International Standard.

ODF has the advantage of being the default format for the OpenOffice (version 2.0 and above). OpenOffice is is a free, open source office suite and can export files to many different formats, including MS Word, rich text format, HTML, PDF and others. Other applications, such as Koffice, use the Open Document Format. Also, Microsoft has recently indicated that it will host an open source project converting documents to ODF from within Microsoft Office.

If OpenOffice documents were to support `Sweave` tags, reports could be edited and formatted in a sophisticated GUI environment. This would allow R users to include `Sweave` tags without directly editing the underlying markup. This would eliminate the two previously discussed limitations related to using LaTeX and HTML.

The **odfWeave** package was created so that the functionality of Sweave can used to generate documents in the Open Document Format. The package is currently limited to creating text documents using OpenOffice. Processing of presentations and spreadsheets should be considered to be experimental (but should be supported in subsequent versions of **odfWeave**).

Getting Started with odfWeave

First, install the most recent version of **odfWeave** using

```
install.packages("odfWeave")
```

at the R command prompt. Once the package is installed, there is a sub-directory called 'examples' that contains several prototype text documents files (with the extension ODT).

Second, since ODF files are compressed archives of files and directories, the user may need to download tools to extract and re-package the files. By default, odfWeave uses the zip and unzip tools which are free and can be found at http://www.info-zip.org/. Other applications, such as jar, can also be used. See the manual for the odfWeaveControl function for more information on using other utility programs.

odfWeave uses almost the exact same syntax as Sweave via LaTeX. Examples of common Sweave tags are illustrated in the next few sections. Once an ODF file is created containing Sweave tags, the document can be processed in R using:

```
odfWeave(inFile, outFile, workDir, control)
```

where inFile and outFile are the source and destination file names, workDir is an optional path where the files are processed and control is an optional object that can be used to specify how odfWeave runs. While the functionality of odfWeave is described in more detail in the next three sections, Figures 1 and 2 have screenshots of ODT files from OpenOffice before and after Sweave processing.

In–line Tags

The simplest way to start working with Sweave is to use in-line tags. These tags can be used to write single lines of text into the document. R code can be encapsulated in the tag \Sexpr{}. For example, putting the line

```
   There were data collected on
   \Sexpr{length(levels(iris$Species))}
   iris species.
```

in a document would produce the output:

```
   There were data collected on 3 iris species.
```

One interesting feature of using odfWeave is the preservation of the text formatting. If all the characters of an in-line Sweave tag have the same format, the formatting carries over into the results. Figures 1 and 2 show an example of formatting that persists. Note that partial formatting of the tag may not affect the output or might generate an error.

Code Chunks

Code chunks are used in much the same way as Sweave with LaTeX. Once exception is that code chunks must use the <<>>= convention (i.e \begin{Scode} is not currently supported).

For example, several R packages produce output on initial startup. To avoid sending this information into the document, a code chunk similar to

```
<<loadLibs, results = hide, echo = FALSE>>=
library("randomForest")
@
```

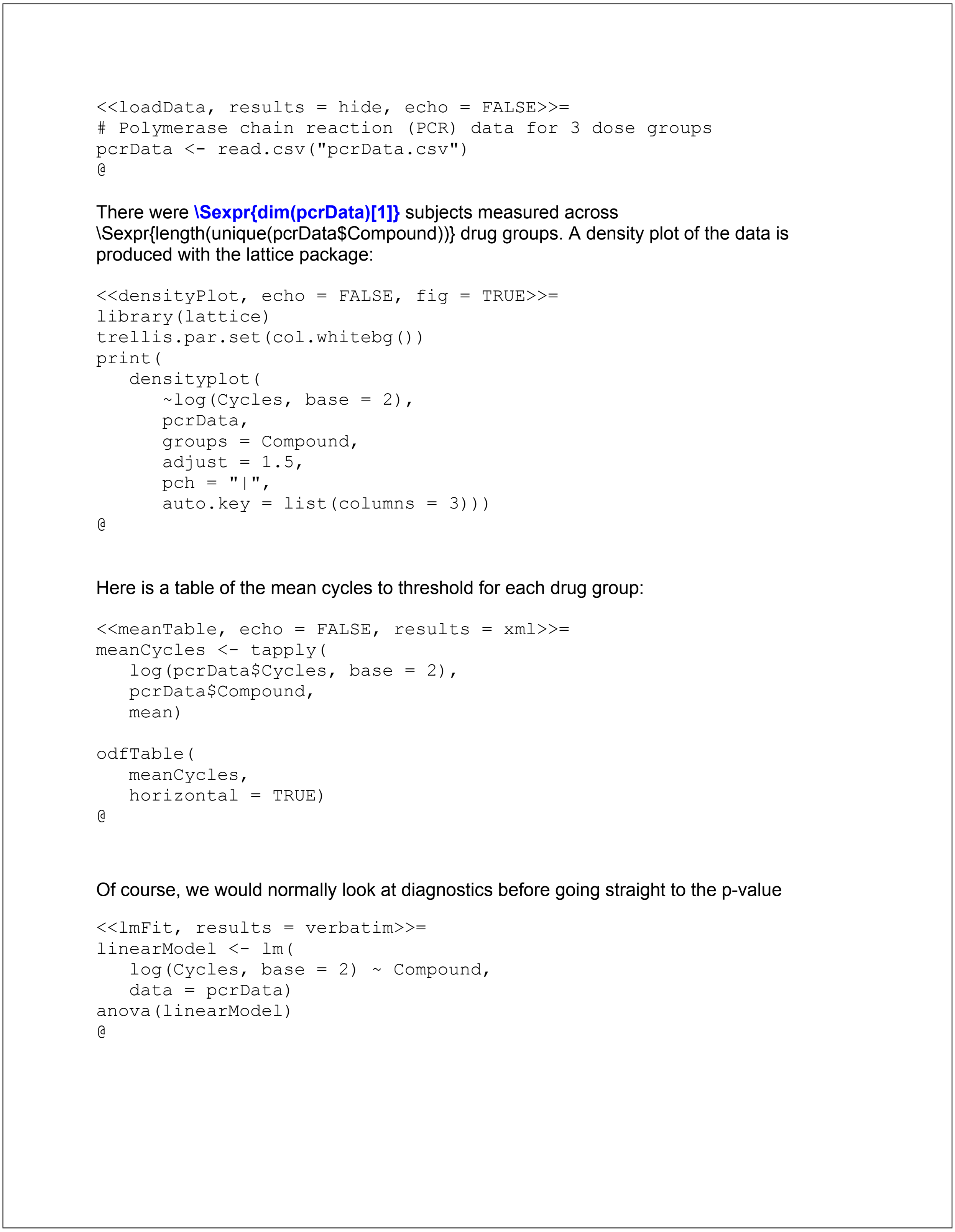
can be used to initialize the package. The first argument provides an optimal label for the chunk. The results argument controls the destination of the output. A value of hide prevents any output from being sent to the document, whereas a value of xml is used when the code chunk generates output that is formatted in the Open Document format (see the next section for an example). Using results = verbatim formats the output of the chunk similar to how it would appear when printed at the command line. The echo argument specifies whether the R code should also be printed in the output.

These, and other options for code chunks in odfWeave are documented with the Sweave driver function, called RweaveOdf. There are some differences in options when compared to Sweave with LaTeX. For example, the odfWeave has more image file format options when compared to those available using LaTeX, so the eps and pdf arguments do not exist for odfWeave.

Tables

Tables can be created from vectors, matrices or data frames in R. The odfTable class can be used in code chucks to create a table in ODF format. For example, to create a table containing the iris data, the code chunk might look like:

```
<<irisTable, results = xml>>=
   # create nice column labels for illustration
   irisCols <- gsub("\\.", " ", names(iris))
   odfTable(
      iris,
      useRowNames = FALSE,
      colnames = irisCols)
@
```



```
<<loadData, results = hide, echo = FALSE>>=
# Polymerase chain reaction (PCR) data for 3 dose groups
pcrData <- read.csv("pcrData.csv")
@
```

There were **\Sexpr{dim(pcrData)[1]}** subjects measured across \Sexpr{length(unique(pcrData$Compound))} drug groups. A density plot of the data is produced with the lattice package:

```
<<densityPlot, echo = FALSE, fig = TRUE>>=
library(lattice)
trellis.par.set(col.whitebg())
print(
   densityplot(
      ~log(Cycles, base = 2),
      pcrData,
      groups = Compound,
      adjust = 1.5,
      pch = "|",
      auto.key = list(columns = 3)))
@
```

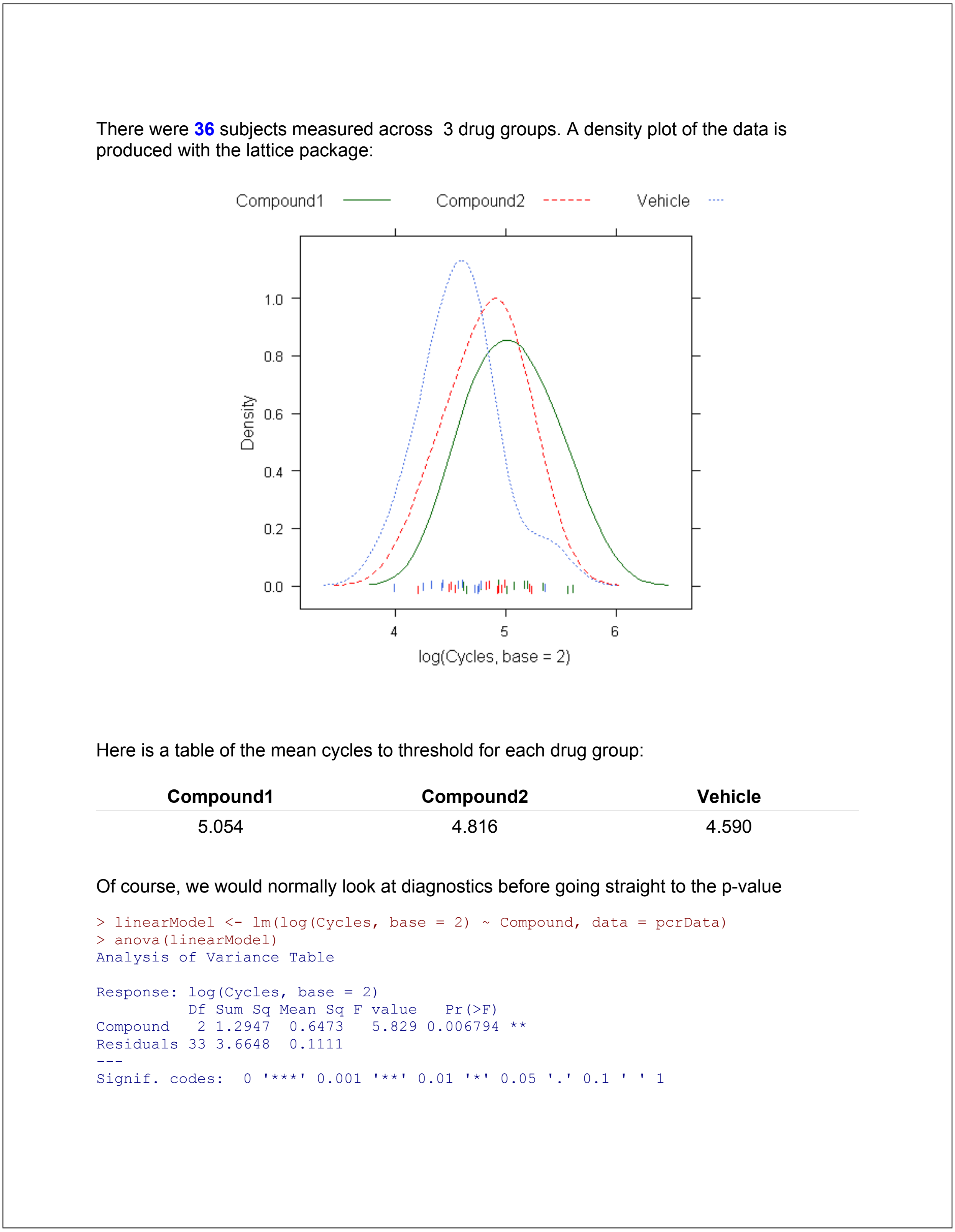
Here is a table of the mean cycles to threshold for each drug group:

```
<<meanTable, echo = FALSE, results = xml>>=
meanCycles <- tapply(
   log(pcrData$Cycles, base = 2),
   pcrData$Compound,
   mean)

odfTable(
   meanCycles,
   horizontal = TRUE)
@
```

Of course, we would normally look at diagnostics before going straight to the p-value

```
<<lmFit, results = verbatim>>=
linearModel <- lm(
   log(Cycles, base = 2) ~ Compound,
   data = pcrData)
anova(linearModel)
@
```

Figure 1: An example of an ODF document containing `Sweave` tags as viewed in OpenOffice.



There were **36** subjects measured across 3 drug groups. A density plot of the data is produced with the lattice package:

Here is a table of the mean cycles to threshold for each drug group:

Compound1	Compound2	Vehicle
5.054	4.816	4.590

Of course, we would normally look at diagnostics before going straight to the p-value

```
> linearModel <- lm(log(Cycles, base = 2) ~ Compound, data = pcrData)
> anova(linearModel)
Analysis of Variance Table

Response: log(Cycles, base = 2)
          Df Sum Sq Mean Sq F value   Pr(>F)
Compound   2 1.2947  0.6473   5.829 0.006794 **
Residuals 33 3.6648  0.1111
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 2: The processed ODF document as viewed in OpenOffice.

First, note that the results argument is set to xml since odfTable will create ODF markup for the table. To illustrate some of the options in odfTable, the column names for the data frame are altered (e.g., "Sepal.Length" becomes "Sepal Length") for the table headings. The argument useRowNames prevents extra column of row names from being created (if useRowNames were set to TRUE (the default), we would have had to create an extra entry in irisCols to account for the column of row names).

Tables for vectors and matrices are handled similarly. See the documentation for odfTable for the details (by typing ?odfTable at the command prompt).

Images

Figures can be created with code chunks by using the fig = TRUE argument. Using the iris data as an example again, we can create a lattice scatterplot matrix of the iris measurements with distinct symbols for each species using

```
<<irisPlot, fig = TRUE>>=
   trellis.par.set(col.whitebg())
   out <- splom(iris[,-5],
      groups = iris$Species,
      pscales = 0,
      auto.key = list(columns = 3))
   # remember, with lattice graphics, we
   # must explicitly print the plot object
   print(out)
@
```

This creates the image file, copies it to the appropriate directory and inserts required markup in place of this code chunk.

The default image format produced is png, although there are many options within OpenOffice. Figure 3 is a screenshot of the available formats. The image format can be specified in the function getImageDefs and setImageDefs. There are options for the image file extension and driver. For example, to specify postscript graphics, the type attribute would be set to "eps" and the device attribute would be set to "postscript". Also, the image file size can be controlled independently of the size of the image as displayed in the document. The image attributes can be changed throughout the document.

As with regular Sweave, Sweavehooks can be used to set graphical parameters before code chunks.

If the need arises to insert a pre–existing image, the odfInsertPlot can be used from within a code chunk:

```
<<insertImage, echo = FALSE, results = xml>>=
   imageMarkup <- odfInsertPlot(
      "myImage.gif",
      height = 2.7,
      width = 2.8,
      externalFile = TRUE)
   cat(imageMarkup)
@
```

This function will copy the image from the original location to the appropriate destination, but unlike odfTable, the odfInsertPlot function doesn't automatically print the output and cat must be used.

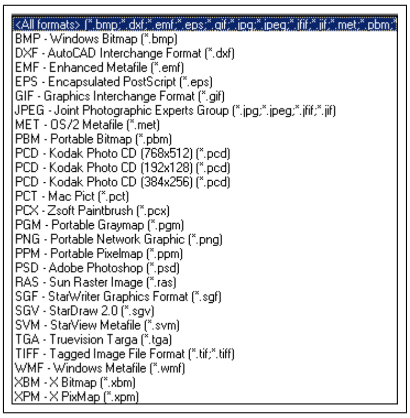
Formatting the Output

There are two main components to specifying output formats: style definitions and style assignments. The definition has the specific components (such as a table cell) and their format values (e.g., boxed with solid black lines). The function getStyleDefs can fetch the pre–existing styles in the package. For example:

```
> getStyleDefs()$ArialNormal
$type
[1] "Paragraph"

$parentStyleName
[1] ""

$textAlign
[1] "center"

$fontName
[1] "Arial"

$fontSize
[1] "12pt"

$fontType
[1] "normal"

$fontColor
[1] "#000000"
```

These can be modified and new definitions can be added. The function setStyledefs "registers" the style changes with the package. When odfWeave is called, these definitions are written to the style sections of the XML files. There is a second mechanism to assign styles to specific output elements. The functions getStyles and setStyles can be used to tell odfWeave which style definition to use for a particular output:

```
> currentStyles <- getStyles()
> currentStyles
$paragraph
[1] "ArialNormal"

$input
[1] "ttRed"

$output
[1] "ttBlue"
```



<All formats> (*.bmp;*.dxf;*.emf;*.eps;*.gif;*.jpg;*.jpeg;*.jfif;*.jif;*.met;*.pbm;
BMP - Windows Bitmap (*.bmp)
DXF - AutoCAD Interchange Format (*.dxf)
EMF - Enhanced Metafile (*.emf)
EPS - Encapsulated PostScript (*.eps)
GIF - Graphics Interchange Format (*.gif)
JPEG - Joint Photographic Experts Group (*.jpg;*.jpeg;*.jfif;*.jif)
MET - OS/2 Metafile (*.met)
PBM - Portable Bitmap (*.pbm)
PCD - Kodak Photo CD (768x512) (*.pcd)
PCD - Kodak Photo CD (192x128) (*.pcd)
PCD - Kodak Photo CD (384x256) (*.pcd)
PCT - Mac Pict (*.pct)
PCX - Zsoft Paintbrush (*.pcx)
PGM - Portable Graymap (*.pgm)
PNG - Portable Network Graphic (*.png)
PPM - Portable Pixelmap (*.ppm)
PSD - Adobe Photoshop (*.psd)
RAS - Sun Raster Image (*.ras)
SGF - StarWriter Graphics Format (*.sgf)
SGV - StarDraw 2.0 (*.sgv)
SVM - StarView Metafile (*.svm)
TGA - Truevision Targa (*.tga)
TIFF - Tagged Image File Format (*.tif;*.tiff)
WMF - Windows Metafile (*.wmf)
XBM - X Bitmap (*.xbm)
XPM - X PixMap (*.xpm)

Figure 3: A screenshot of available image formats in OpenOffice (version 2.0).

```
$table
[1] "Rtable1"

$cell
[1] "noBorder"

$header
[1] "lowerBorder"

$cellText
[1] "ArialCentered"

$headerText
[1] "ArialCenteredBold"

$bullet
[1] "Rbullet"
```

For example, the input and output elements control how R code and command–line output look. To change either of these, an existing definition can be assigned to these entries and reset using setStyles(currentStyles). Unlike the style definitions, the style assignments can be modified throughout the R code (although only the style definitions available when odfWeave is called can be used).

The package also contains a function, tableStyles, that can be used to differentially format specific cells and text in tables.

Other Functionality

odfWeave can also be used to created bulleted lists from vectors via the odfItemize and formatted text can be written using odfCat.

There are a few miscellaneous functions also included in the package. pkgVersions can create a summary of the packages and versions in the search path. The function listString takes a vector and returns a textual listing. For example,

```
listString(letters[1:4])
```

would become "a, b, c and d". Also, matrixPaste can take a series of character matrices with the same dimensions and perform an element–wise paste.

Summary

odfWeave provides functionality for weaving R into documents without dealing with the details of the underlying markup language. The usage of odfWeave closely mirrors that of standard Sweave so current Sweave users should be able to move between the two formats easily.

Appendix: Under the Hood of an ODF File

Open Document Format files are compressed archives of several files and folders which can be decompressed using standard tools such as unzip, jar or WinZip. A typical document will have a structure similar to:

```
Name
----
content.xml
layout-cache
META-INF/
META-INF/manifest.xml
meta.xml
mimetype
Pictures/
Pictures/rplot.png
settings.xml
styles.xml
Thumbnails/
Thumbnails/thumbnail.png
```

The main points of interest are:

- 'content.xml' contains the text, tables, lists and any other user–supplied content. It contains limited formatting.

- ‘style.xml’ contains most of the formatting definitions, including fonts, table formats, and page layout.
- the ‘Pictures’ directory contains any images that were inserted into the document. When images are added, the image files are saved in the archive in their native format

Sweave tags inserted into the body of a document would primarily effect ‘content.xml’.

OdfWeave requires a zip/unzip utility to gain access to these files, process them and re-assemble the ODT file so that OpenOffice can re-open it.

Bibliography

F. Leisch. Sweave, Part I: Mixing R and LaTeX. *R News*, 2(3):28–31, 2002. URL http://cran.r-project.org/doc/Rnews.

F. Leisch. Sweave, Part II: Package Vignettes. *R News*, 3(2):21–34, 2003. URL http://cran.r-project.org/doc/Rnews.

Organization for the Advancement of Structured Information Standards (OASIS). *Open Document Format for Office Applications (OpenDocument)* v1.0. May, 2005. URL http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf.

Max Kuhn
Research Statistics
Pfizer Global Research and Development
max.kuhn@pfizer.com

plotrix

A package in the red light district of R

by Jim Lemon

The social science of statistics

It is firmly established that statistics can assist the deceptive, but less well recognized that it can also aid those practising one of the other ancient arts. Some statisticians do attain the position of courtesans to the mighty, where they may spurn the lowlier tasks of their trade. Most of us must entertain the whims of the paying customer.

Like many R users, I did not aspire to be a statistician. The combination of a bit of relevant talent and the disinclination of others led to my recruitment. Readers are probably well aware that legions of reluctant statisticians desperately scan their sometimes inadequate knowledge in the effort to produce what others with considerably less knowledge demand.

In this spirit I began to program the functions that were included in the initial version of **plotrix** (Lemon, 2006). Inspiration was found not only in my own efforts to use R as I had previously used other statistical or graphical packages, but in the pleas of others like me that found their way to the R help list. Commercial software has lent a sympathetic, if not always disinterested, ear to such requests. You want a 3D pie chart with clip art symbols floating in the transparent sectors and you get it, for a price. There is a strong desire for the conventional in the average consumer of statistics, even if the conventional is not the most informative.

I would like to introduce **plotrix** using three topics in plotting that have been the subject of some discussion on the R help list as well as a little parade of the functions.

How to slice the pie

Consider the aforementioned pie chart. There have been several discussions upon the wisdom of using such illustrations, typically initiated by a query from an R user. Despite the limitations of the pie chart, they seem common. To test this impression, I asked our friend Google to inform me on the phrase “division of wealth” and then stepped through the listing until I found an illustration of this phrase. On the twelfth hit, there was ... a pie chart (Ng, 2006). As an example of the way in which the **plotrix** package may be used, I reproduced it in Figure 1. The code I used is shown below.

```
x11(width=7, height=4)
par(mar=c(0,0,0,0), xpd=FALSE)
plot(0, xlim=c(-2,2.2), ylim=c(-1,1),
    xlab="", ylab="", type="n")
rect(-2.5, -1.5, 2.5, 1.5, col="#ffffcb")
wealth.pct<-c(41, 15, 9.4, 6.4, 4.8, 3.7,
             2.9, 2.4, 2, 1.6, 12.1)
sector.colors<-c("#fe0000", "#ffff00", "#0000fe",
                 "#ff00fe", "#00ffff", "#fc6866",
                 "#666632", "#d2ffff", "#fdffcd",
                 "#ffcb65", "#ffffff")
floating.pie(0, 0, wealth.pct, col=sector.colors,
            startpos=pi/2)
text(-1.55, 0.2,
    "wealth owned by the\nrichest one percent
of the population")
text(-1.4, -0.9,
    "wealth owned by the second
richest one percent")
segments(-0.65, -0.85, -0.3, -0.8)
text(1.3, -0.7, "by third richest percentile")
text(1.55, -0.45, "by fourth richest percentile")
text(1.65, -0.1, "by fifth richest percentile")
text(1.64, 0.12, "by sixth richest percentile")
text(1.66, 0.3, "by seventh richest percentile")
text(1.55, 0.43, "by eighth richest percentile")
text(1.47, 0.53, "by ninth richest percentile")
text(1.37, 0.62, "by tenth richest percentile")
text(1, 0.95,
"Wealth left to the remaining 90% of the
 population")
text(1.3, 0.85,
    "(those of us worth less than $8 million)")
segments(0.2, 0.9, 0.23, 0.85)
par(cex=0.7)
text(1, -0.95,
"Copyright 2001 by Freeman Ng and
 Progressive Schoolhouse")
text(1, -1.02,
    "(http://www.ProgressiveSchoolhouse.org)")
par(cex=1.5)
text(-2, 0.9, "How the Pie is Sliced", adj=0)
par(cex=1, mar=c(5, 4, 4, 2), xpd=TRUE)
```

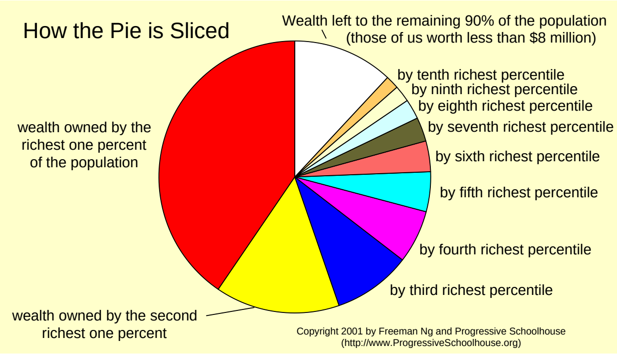


Figure 1: A reproduction of a pie chart

The R user can thus produce a reasonable approximation of this rather complicated chart. However, the beauty of the `floating.pie` function is not solely its ability to copy other people's work. This chart is a good illustration of the major complaint about pie charts, our limited ability to translate the areas of circular sectors into relative quantities. The intent of the author was obviously to illustrate the concentration of private wealth in the USA in a minority. Compare the next chart, showing the relative wealth of the richest one percent, the rest of the top half and the poorest half of the population. Now it is easy to see that the richest one percent hold nearly half the wealth, the rest of the top half a bit more, and only a tiny slice is shared amongst the poorest half. The much more compact code is again below the figure.

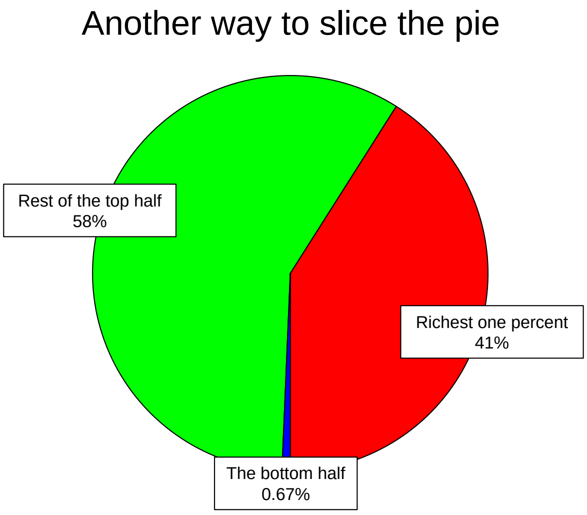


Figure 2: A reworked pie chart

```
par(xpd=NA, mar=c(2, 2, 2, 2))
plot(0, xlim=c(-1, 1), ylim=c(-1, 1), xlab="",
    ylab="", axes=FALSE, type="n")
wealth.pct <- c(41, 58.33, 0.67)
bisectors <- floating.pie(0, 0, wealth.pct,
                         radius=0.8,
                         startpos=3*pi/2)
pie.labels(0, 0, bisectors,
          c("Richest one percent\n41%",
            "Rest of the top half\n58%",
            "The bottom half\n0.67%"),
          radius=0.85, ypad=0.8)
par(cex=2)
text(0, 1, "Another way to slice the pie")
par(xpd=TRUE, mar=c(5, 4, 4, 2), cex=1)
```

It is clear that a pie chart with a small number of sectors can present a “big slice-little slice” distribution as a “sight bite.” `floating.pie` is intended to overlay small pie charts on other plots. Thus one could add a series of “top half/bottom half” pie charts to a plot of GNP by country to show how the distribution of wealth is related to GNP.

How to bridge the gap

While `floating.pie` was only a modification of the existing `pie` function, many of the other plot variants were programmed from the bottom up, such as `pie3D`. Another vexed area in R is how to deal with empty space. When the values to be plotted are all very large or small, R will typically display only the range of those values. However, if the values fall into two groups, we will see the smaller values at the bottom, a large gap, and the larger values at the top of the resulting plot. The initial response to the former problem was the `axis.break` function. This displayed the conventional slash or zigzag break symbol on the axis chosen by the user.

When addressing the problem of values that fall into two groups, it was clear that the conventional symbols might not be sufficient to indicate a discontinuous axis. Consider the challenge of plotting the heights of the three highest mountains in Australia, a notably flat continent, and the three highest in Nepal. The figure below shows how gap.plot handles this, omitting a 6100 meter gap that would take up more room than the entire range of heights displayed. gap.plot is also useful for displaying outliers. As it is of general use, the "gap" axis break was added to the axis.break function.

staxlab, another function from **plotrix**, came to the rescue of the mountain names, as the "intelligent clipping" of the x-axis labels snipped two off. To fix this, empty names were passed, and then the labels were put in place. They didn't really need to be staggered, but it was a good opportunity for staxlab to show off.

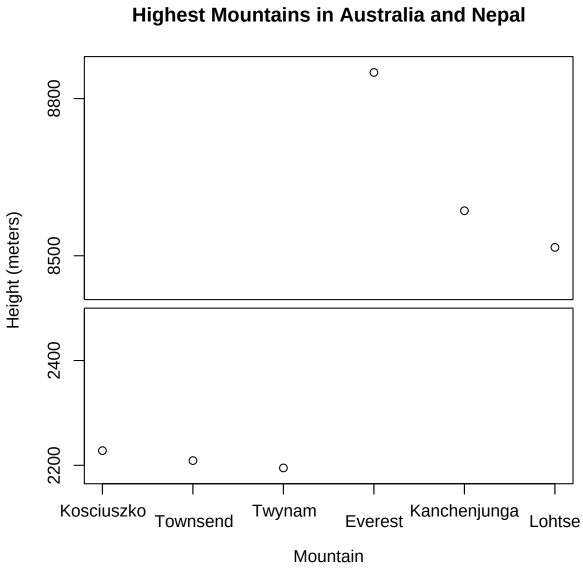


Figure 3: Australia vs Nepal

```
mountains<-c(2228, 2209, 2195, 8850, 8586, 8516)
gap.plot(mountains, gap.bounds=c(2500, 8400),
        gap.axis="y",
        xaxlab=c("","","","","",""),
        xtics=1:6,
        ytics=c(2200, 2400, 8500, 8800),
        main=
        "Highest Mountains in Australia and
 Nepal",
        xlab="Mountain", ylab="Height (meters)")
staxlab(at=1:6,labels=c("Kosciuszko","Townsend",
                        "Twynam","Everest",
"Kanchenjunga","Lohtse"),
        nlines=2,top.line=0.8,line.spacing=0.5)
```

How to appear 3D

Another topic that frequently appears on the R help list is the wisdom of trying to get more than two dimensions into a two dimensional illustration. I'll use this to introduce triax.plot and its relatives. This function was originally programmed in response to a request for a "soil texture" plot in which the proportions or percentages of three arbitrary particle size categories are displayed. This is a particular example of the more general "triangle" plot. In response to another request, the function was completely rewritten as triax.plot and soil.texture became a restricted call to this function. Other capabilities were added, such as triax.abline allowing lines representing values on the axes to be added. While the triangle plot appears to show three value coordinates, it only works because the coordinates are not independent. The triplets must sum to 1 for proportions or 100 for percentages. triax.plot will output a warning if this condition is not satisfied. In contrast, triangle.plot in **ade4** (Penel, 2005) silently transforms each set of values to percentages. This may not be what the user intended in some cases, so triax.plot leaves the decision to transform to the user. The somewhat odd name is to avoid confusion with other triangle plot functions.

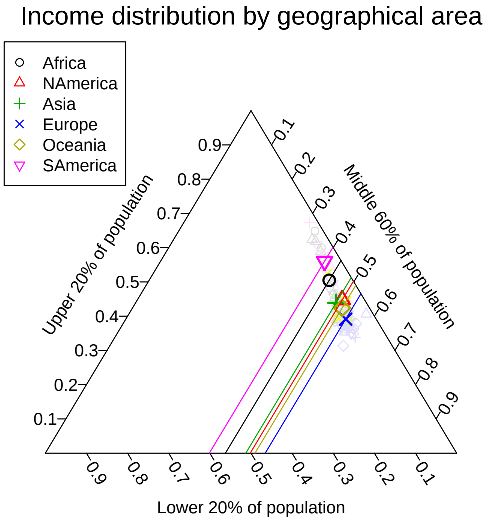


Figure 4: Income distribution in geographical areas

To demonstrate triax.plot, we'll return to the evergreen subject of money. After locating a reasonably authoritative dataset on income inequality (Deininger, 2005) the most recent and highest quality entries for 80 countries were extracted. The countries were grouped into six geographical areas, and the cumulative income quintiles were collapsed into three categories, lowest 20%, middle 60% and upper 20%. contavg contains the average income distributions for the six geographical areas. If everyone in an area earned the same income, the point would lie in the lower right corner of the triangle at (0.2,0.6,0.2). If the top dogs or fat cats got almost everything, the point would be at the top of the pyramid. The closer the point is to the right axis, the worse the poorest 20% are doing. After displaying all the individual countries in a faded set of colors, the averages were overlaid in stronger colors and larger symbols for a comparison. Being a member of the middle class, I then added lines to emphasize the relative position of this group in the areas defined. While `triax.plot` is not ideal for illustrating this particular sort of data, this example shows the capabilities of the function. The data file can be obtained by emailing the author.

```
incdist.df<-
 read.csv("/home/jim/R/doc/incdist.csv")
incprop<-cbind(incdist.df$Low20,
               incdist.df$Mid60,
       incdist.df$Hi20)
dark.col<-c("#000000","#ff0000","#00aa00",
            "#0000ff","#aaaa00","#ff00ff")
faded.col<-c("#dddddd","#ffdddd","#ddffdd",
             "#ddddff","#ffff00","#ffddff")
triax.plot(incprop,
 main="Income distribution by geographical area",
 axis.labels=c("Lower 20% of population",
               "Middle 60% of population",
               "Upper 20% of population"),
 col.symbols=
  faded.col[as.numeric(incdist.df$Area)],
 pch=1:6)
areaavg<-cbind(
 as.vector(
  by(incdist.df$Low20,incdist.df$Area,mean)),
 as.vector(
  by(incdist.df$Mid60,incdist.df$Area,mean)),
 as.vector(
  by(incdist.df$Hi20,incdist.df$Area,mean)))
triax.points(areaavg,col.symbols=dark.col,
 pch=1:6,cex=1.5,lwd=2)
triax.abline(r=areaavg[,2],col=dark.col)
legend(-0.1,1.04,
 legend=c("Africa","NAmerica","Asia",
          "Europe","Oceania","SAmerica"),
 pch=1:6,col=dark.col)
```

Below is a list of most of the functions in **plotrix** v2.1, which should be the version available when you are reading this. Functions usually called by another function are not included.

```
axis.break
 Put a "break" in a plot axis
axis.mult
 Display an axis with a multiplier
barplot.symbol.default
 Barplot with bars filled with symbols
bin.wind.records
 Classify wind direction and speed records
boxed.labels
 Display text strings in boxes
clean.args
 Remove arguments that would cause an error
clock24.plot
 Plot values on a 24 hour "clockface"
cluster.overplot
 Display overlying points as clusters
color.id
 Find the closest color name to an RGB code
color.scale
 Transform numeric values into colors
color2D.matplot
 Display a numeric matrix as a color matrix
corner.label
 Display a label in one corner of a plot
count.overplot
 Display overlying points as a number
feather.plot
 Display vectors along a horizontal line
floating.pie
 Flat pie chart at a specified position
freq
 Calculate frequencies
gantt.chart
 Gantt chart with task priority colors
gap.barplot
 Barplot with a range of values omitted
gap.plot
 Plot with a range of values omitted
get.gantt.info
 Enter information for a Gantt chart
get.soil.texture
 Enter information for soil textures
get.triprop
 Enter triplets of proportions or percentages
multhist
 Plot a multiple histogram as a barplot
multsymbolbox
 Boxes filled with symbols representing counts
oz.windrose
 Australian BOM wind rose
pie3D
 Display a 3D pie chart
plot.freq
 Horizontal barplot of frequencies
plotCI
 Display confidence intervals/error bars
polar.plot
 Plot values on a 0-360 degree scale
polygon.shadow
 Display a shadow effect for a polygon
print.freq
 Display frequency tables
radial.plot
 Plot values on a 0-2*pi radian scale
remove.args
 Remove specified arguments from a list
sizeplot
 Display overlying points as larger points
smoothColors
 Build a vector of interpolated colors
soil.texture
 Display a "soil texture triangle"
spread.labels
 Spread out labels for values
staxlab
 Display staggered labels on an axis
```

```
symbolbox
 Draw a box filled with symbols
textbox
 Display justified text in an optional box
thigmophobe.labels
 Place labels away from the nearest other point
triax.plot
 Display a triangle (three axis) plot
```

The proliferation of packages and functions for R has led to many similar functions in different packages. The `radial.plot` family, including `radial.plot`, `polar.plot` and `clock24.plot` has relatives including `rosavent` in **climatol**, (Guijarro, 2004) `plot.circular` in **circular** (Agostinelli, 2005) and `rose.diag` in **CircStats** (Agostinelli, 2005). The presence of similar functions in different packages that give the user more choices leads to the ensuing difficulty of finding the functions that afford the choices. There have been several attempts to minimize the latter effect. My own favorite is arbitrary text searching such as that implemented by Professor Jonathon Baron's R Site Search facility (Baron, 2006).

A parting glance at style

plotrix has a strong collaborative influence, and it is hoped that this will continue. The programming style leans toward the explicit rather than being highly condensed or efficient, and many R users have contributed not only suggestions but segments of code that have been incorporated. Those experienced in the more arcane techniques of illustration have offered valuable comments on improving certain functions. The overall utility of **plotrix** is largely a product of this expert feedback. A possible extension of **plotrix** would be to add compatibility with the more flexible **grid** package plotting functions.

The aim of **plotrix** is not to erode the high standards of R but to allow its users to perform the more mundane tasks of their calling with less effort. The consumer who accepts that R can produce the comforting pictures to which he or she is accustomed may well be seduced into the more challenging illustrations of data. It is hoped that **plotrix** and similar packages will provide functions that allow the new user to demonstrate the basic capabilities of R as rapidly as any other statistical software and the experienced user to generate the common varieties of data illustration more conveniently.

Bibliography

C. Agostinelli. *circular: Circular statistics.* URL `http://cran.r-project.org/src/contrib/Descriptions/CircStats.html`

C. Agostinelli. *CircStats: Circular statistics.* URL `http://cran.r-project.org/src/contrib/Descriptions/circular.html`

J. Baron. *R Site Search* URL `http://finzi.psych.upenn.edu/search.html`

K.W. Deininger. *Measuring Income Inequality Database.* URL `http://siteresources.worldbank.org/INTRES/Resources/469232-1107449512766/648083-1108140788422/A_New_Dataset_Measuring_Income_Inequality.zip`

J.A. Guijarro. *Climatol: some tools for climatology.* URL `http://cran.r-project.org/src/contrib/Descriptions/climatol.html`

J. Lemon. *plotrix: Various plotting functions.* URL `http://cran.r-project.org/src/contrib/Descriptions/plotrix.html`

F. Ng. *The Division of Wealth* URL `http://www.progressiveschoolhouse.org/wealth/wealthdivisions.htm`

S. Penel. *ade4: Multivariate data analysis and graphical display.* URL `http://cran.r-project.org/src/contrib/Descriptions/ade4.html`

Jim Lemon
bitwrit software, Gladesville, Australia
`jim@bitwrit.com.au`

rpanel: making graphs move with tcltk

by Adrian Bowman, Ewan Crawford, and Richard Bowman

The command line interface of R provides a flexibility and precision which is very well suited to many settings. Alternatively, at the introductory level, where point-and-click interfaces can be useful, *gui* interfaces such as **R Commander** described by Fox (2005) are available. However, in between these two modes of use there is sometimes a very useful role for interactive control of R operations, particularly where graphics are involved. The **tcltk** package of Dalgaard (2001), which provides a link from R to the *Tcl/Tk* system, helpfully provides a very extensive set of tools for this purpose and this system has featured regularly in the pages of *R News*.

The aim of the **rpanel** package is to provide a simple set of tools such as buttons, sliders, checkboxes and textentry boxes, each of which can be created by a single function call, with full documentation. The package essentially provides a set of wrappers for underlying **tcltk** operations which are intended to make access to these controls as simple as possible. In particular, the **tcltk** variables which are the basis of communication between the control panel and R are managed behind the scenes in a way that is transparent to the user. Some simple facilities for interacting with images are also provided.

Tools for interactive controls within R is a rapidly developing area, with some exciting prospects. The **iplots** package of Urbanek and Theus (2003) offers genuine interaction with Java-based plots, while the **RGtk2** package of Lang and Lawrence provides a set of gui building-blocks that is similar to **tcltk**. The main emphasis of the **rpanel** package is to provide a simple and easy-to-use set of controls which can enhance the standard R plots which form familiar territory to the majority of R users.

A simple example

A simple example is provided by the application of a Box-Cox power transformation in the context of a Q-Q plot for normality. Tierney (1990) used this as an illustration of the dynamic graphics available in the Lisp-Stat system. The R code below defines a power transformation function `bc.fn` and a simple plotting function `qq.draw` which applies the transformation and passes the result to `qqnorm`. The final two statements set up a control panel and a slider. Movement of the slider causes the 'action' function `qq.draw` to be called with the current setting of the variable `lambda`. This simply redraws the plot to create an animation effect.

```
bc.fn <- function(y, lambda) {
   if (abs(lambda) < 0.001) z <- log(y)
   else z <- (y^lambda - 1)/ lambda
   }
qq.draw <- function(panel) {
   z <- bc.fn(panel$y, panel$lambda)
   qqnorm(z, main = paste("lambda =",
          round(panel$lambda, 2)))
   panel
   }
panel <- rp.control(y = exp(rnorm(50)),
                    lambda = 1)
rp.slider(panel, lambda, -2, 2, qq.draw)
```

An illustration is given in Figure 1, where the lower plot shows the untransformed data (`lambda` = 1) and the upper plot show the effect of a log transformation. Since the data were simulated as exponentially transformed normal random variables, this is the appropriate transformation back to normality. Here `qq.draw` has been modified in a simple way to add a histogram of the transformed data in a second panel of the plot. It would be easy to amend the code in other ways, such as drawing the plot in a different colour for values of `lambda` that lie inside the likelihood-based 95% confidence interval for the power transformation parameter.

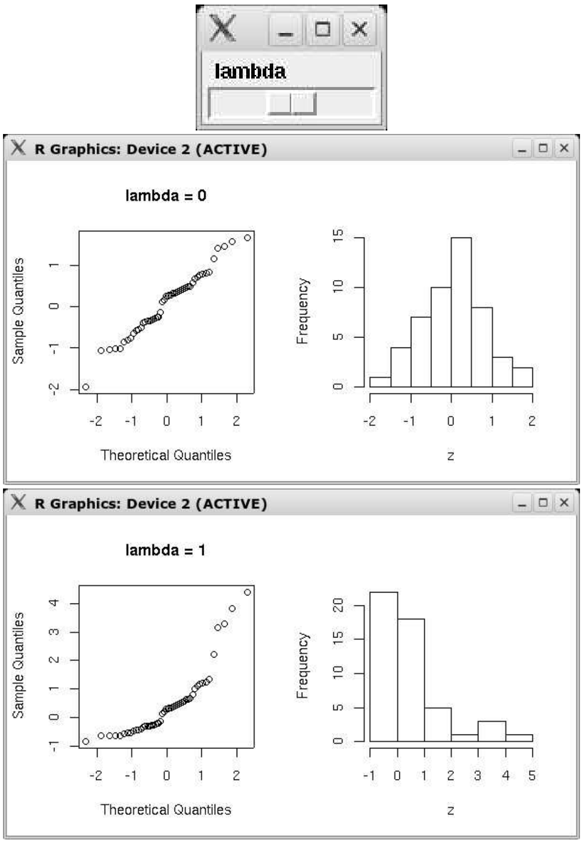


Figure 1: Slider control of a power transformation for a normal q-q plot.

Those who use the **tcltk** package will be entirely familiar with this kind of facility. It should be emphasised again that the principal aim of the **rpanel** package is simply to make access to these facilities as straightforward as possible for those who are less well placed to become familiar with the *Tcl/Tk* system. The user can then direct the majority of attention to the construction of the appropriate action functions. The `panel` object created by the `rp.control` function contains the data to be used by the action functions, including the variables associated with subsequent widgets (buttons, sliders, etc.). An advantage of this construction is that the underlying communication variables can be managed in a manner which allows the use of functions and other multiple instances of the same R code, without creating communication difficulties.

Note that it is important that action functions are defined before the **rpanel** functions which create the panel controls are called. This is because the action functions are attached to the panel object for later execution. It is also important that each action function returns the R panel object. This is a feature of the communication mechanism used by the package. It is clearly particularly relevant in situations where an action function may alter the contents of the panel object and so this needs to be made available for subsequent use.

Further examples

A full description of the **rpanel** package is provided in a paper which is available at `www.stats.gla.ac.uk/~adrian/rpanel/`. A set of illustration scripts, including the code for the examples of this article, can also be found there. One or two further illustrations of the package and its potential uses are given here.

The `persp` function in R allows three-dimensional perspective plots to be created, with the viewing angle controlled by the arguments `theta` and `phi`. It can be helpful to rotate the surface to examine features that are hidden by the current perspective. The code below creates an action function `volcano.plot` which simply draws the graph with the current values of `theta` and `phi`. The use of the `with` environment helpfully allows components of the `panel` object to be referred to directly. Two doublebuttons are then inserted into a control panel. When pressed, the relevant parameter is incremented or decremented by 10 degrees and the plotting function called. With a plotting task of this complexity, the refresh rate is not sufficiently fast to create a very smooth animation. Nonetheless, the interactive control is very useful in visual examination of the surface.

```
z <- 2 * volcano
x <- 10 * (1:nrow(z))
y <- 10 * (1:ncol(z))
volcano.plot <- function(panel) {
  with(panel, {
    persp(x, y, z, theta = theta, phi = phi,
      col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75,
      border = NA, box = FALSE)
    })
  panel
  }
volcano.panel <- rp.control(x = x, y = y,
         z = z, theta = 135, phi = 30)
rp.doublebutton(volcano.panel, theta, 10,
         action = volcano.plot)
rp.doublebutton(volcano.panel, phi,   10,
         action = volcano.plot)
rp.do(volcano.panel, volcano.plot)
```

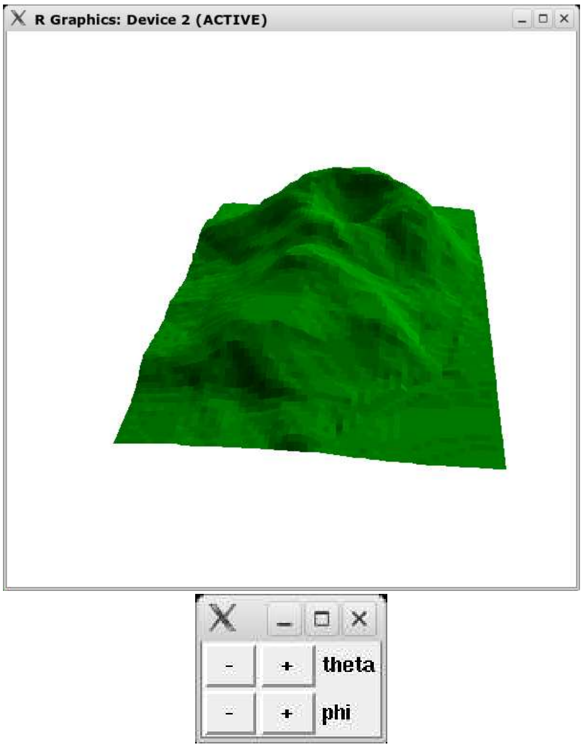
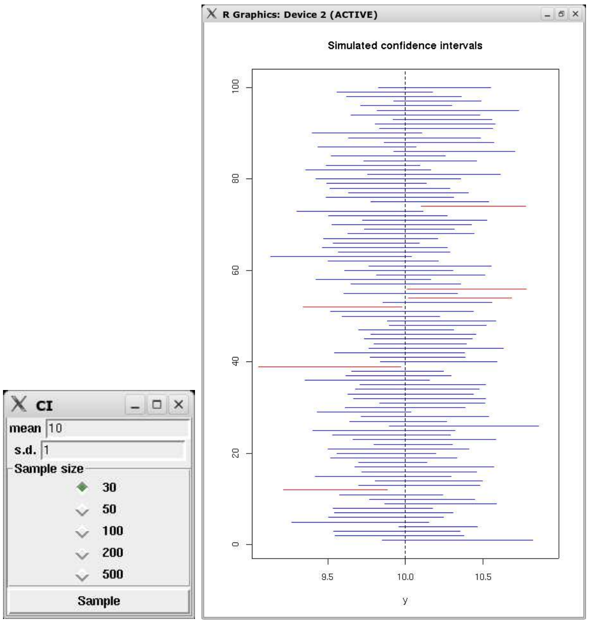


Figure 2: Doublebutton control of the viewing angles of a persp plot.

A further example involves the simulation of confidence intervals. This can be a very useful exercise in a teaching context, to communicate the meaning of ‘confidence’ through the long-run proportion of constructed intervals that contain the true value of the parameter of interest. Figure 3 shows a control panel which contains text-entry boxes to supply the mean and standard deviation of the normal population from which the data are sampled, radiobuttons to specify the sample size and a button to create a set of confidence intervals. The code to create the controls is straightforward and is shown below. The code for the action function is not shown but involves simple plotting procedures which are very familiar to the majority of R users. The teacher is therefore able to create the plot and its controls to suit personal preferences. Here the intervals that miss the true value of the parameter are coloured red.

There would be no difficulty in turning the code for this example into a function by adding the necessary wrapping statements. Repeated calls of the function would create independent control panels with their own communication mechanisms. It would clearly be neater in that case to add an `x11()` statement to launch a new graphics window for each call of the function. The use of `dev.cur` and `dev.set` would then allow the appropriate graphics window to be activated by each control panel, using an identifier contained in the R panel object. The binomial example on the `rpanel` web site gives an example of this.

```
ci.panel <- rp.control("CI", mu = 10,
   sigma = 1,
   sample.sizes = c("30","50","100","200",
   "500"))
rp.textentry(ci.panel, mu,
   title = "mean", action = ci.plot)
rp.textentry(ci.panel, sigma,
   title = "s.d.", action = ci.plot)
rp.radiogroup(ci.panel, ssize,
   c(30, 50, 100, 200, 500),
   title = "Sample size",
   action = ci.plot)
rp.button(ci.panel, title = "Sample",
   action = ci.plot)
```



CI

mean 10

s.d. 1

Sample size

● 30
○ 50
○ 100
○ 200
○ 500

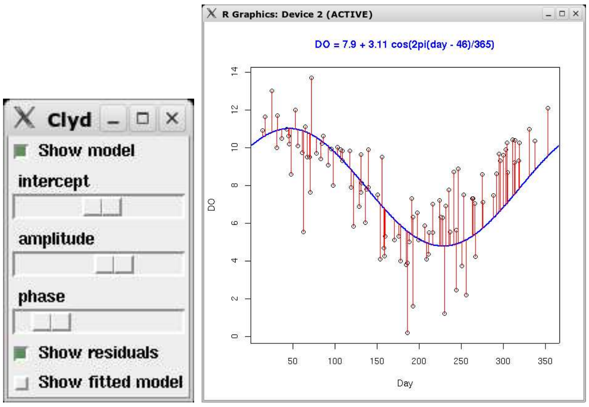
Figure 3: Panel control of simulated confidence intervals.

Again in a teaching setting, it can be very helpful to use graphics to communicate the form of a model, the meaning of its parameters and the process of fitting the model to observed data. Figure 4 shows data on dissolved oxygen (DO) from the River Clyde in Scotland, measured at a particular sampling station over a period of several years. (The data were kindly provided by Dr. Brian Miller of the Scottish Environment Protection Agency.) The data are plotted against day of the year and a clear seasonal pattern is evident. A simple model for this is based on a shifted and scaled cosine curve as

$$y_i = \alpha + \beta \cos((x_i - \gamma)2\pi/366) + \varepsilon_i.$$

In order to illustrate the meaning of this model to students, it is helpful to create a control panel which allows the values of the intercept (α), scale (β) and phase shift (γ) to be altered by sliders. The intuitive meaning of each parameter is then clear. A checkbox has been used to allow the residuals to be displayed, leading to the concept of least squares fitting. A further checkbox allows the fitted model to be shown. (Expansion of the *cos* term in the model shows that it is in fact a linear model, but that is not the central issue in the illustration.)

The code for this example has been omitted. However, it involves a simple plotting function whose display is controlled by the logical flags associated with the checkboxes and the values of the parameters set by the sliders. Each widget in the control panel is created by a single call to the appropriate **rpanel** function. The full code is available on the web site referred to at the end of the article.



Clyd

☑ Show model

☑ Show residuals

☐ Show fitted model

intercept
amplitude
phase
R Graphics: Device 2 (ACTIVE)
DO = 7.9 + 3.11 cos(2pi(day - 46)/365)
DO
0
2
4
6
8
10
12
14
50
100
150
200
250
300
350
Day

Figure 4: Panel control of the parameters of a cosine regression model.

Using images

A more unusual example involves interaction with images. Figure 5 shows a map of the Clyde estuary with a number of sampling points superimposed. (A more detailed high quality map could be used but this rather simple version has been used to avoid copyright issues.) Underneath the map a doublebutton has been inserted to allow movement along the sampling stations. The current station is marked by a yellow cross and the data from that location are plotted in the R graphics window. This allows the changing pattern of overall level and seasonal effect to be examined in an animated manner.

However, it is also possible to click on the map to identify a sampling station and plot the corresponding data. This gives a greater degree of interactive control. The code used for this is shown below. The `clyde.plot` function marks the nominated sampling station and produces the R plot of the corresponding data. The `click.action` function reacts to a click on the map image and checks whether this is close to a sampling station, in which case this is made the current station. The `clyde.plot` function is then called. The `rp.image` function inserts the map image into the control panel while the `rp.clearlines` and `rp.line` functions are used to mark the current sampling station. The detailed use of these functions can be seen from the package documentation.

```
clyde.plot <- function(panel) {
  with(panel, {
    rp.clearlines(panel, clyde.image)
      rp.line(panel, clyde.image,
              station.x[station] - 10,
              station.y[station] - 10,
              station.x[station] + 10,
              station.y[station] + 10,
              width = 4, color = "yellow")
      rp.line(panel, clyde.image,
              station.x[station] + 10,
              station.y[station] - 10,
              station.x[station] - 10,
              station.y[station] + 10,
              width = 4, color = "yellow")
    ind <- (Station == station)
    plot(DO[ind] ~ Day[ind], ylim = range(DO),
         xlab = "Day", ylab = "DO")
    })
  panel
  }
click.action <- function(panel, x, y) {
  x <- as.numeric(x)
  y <- as.numeric(y)
  distance <- sqrt((x - panel$station.x)^2 +
                   (y - panel$station.y)^2)
  if (min(distance) <= 25) {
    panel$station <-
            which(distance == min(distance))
    clyde.plot(panel)
    }
  panel
  }
clyde.panel <- rp.control("Clyde",
  station = 1,
  station.x = c(913, 849, 791, 743, 695, 660,
           600, 555, 485, 407, 333, 249, 167),
  station.y = c(550, 522, 502, 467, 432, 392,
           362, 312, 306, 294, 274, 249, 218),
  Day = Day, Station = Station, DO = DO)
rp.image(clyde.panel, "clyde.gif",
  id = "clyde.image", action = click.action)
rp.doublebutton(clyde.panel, station, 1,
  action = clyde.plot, range = c(1, 13))
rp.do(clyde.panel, clyde.plot)
```

Discussion

The interactive control of R graphics, or indeed R functions more generally, has considerable potential in a variety of settings. One of the main uses is the creation of animation effects through instantaneous redrawing when control widgets are activated. There are many potential applications of this in the graphical exploration of data, in teaching and in the creation of stand-alone tools for particular applications which can be used by those without knowledge of R.

All of the facilities required for these tasks are already available in the **tcltk** package. The aim of the **rpanel** package is to make a useful subset of these as widely available as possible by providing functions to create control widgets in single calls, with full documentation. In addition, the communication mechanism allows functions that create control panels to be written in the usual way. R graphics can also be incorporated into a control panel using the **tkrplot** package of Tierney.

A more extensive description of the package is available at

www.stats.gla.ac.uk/~adrian/rpanel/

where a collection of further examples is also located.

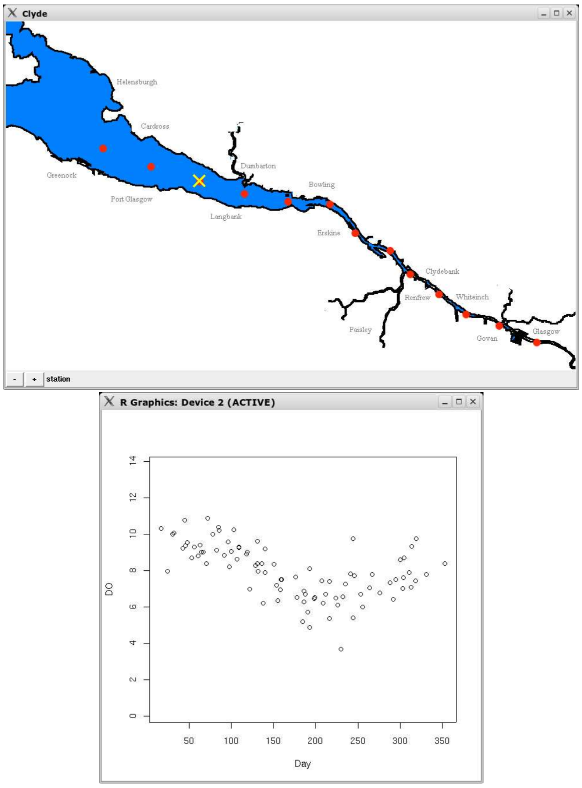


Figure 5: Use of a map to identify the station for which data should be plotted (the marked positions of the sampling locations are approximate, for the purposes of illustration).

Bibliography

P. Dalgaard. The R-tcl/tk interface. In *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*, eds. K.Hornik, F.Leisch. Vienna, March 2001. ISSN: 1609-395X, 2001.

J. Fox. The R commander: a basic-statistics graphical user interface to R. *Journal of Statistical Software*, 14, 2005. URL http://www.jstatsoft.org/.

D. T. Lang and M. Lawrence. **RGtk2**: R bindings for GTK 2.0. URL http://cran.r-project.org/.

L. Tierney. *Lisp-stat: an object-oriented environment for statistical computing and dynamic graphics*. Wiley: New York, 1990.

L. Tierney. **tkrplot**: simple mechanism for placing R graphics in a tk widget. URL http://cran.r-project.org/.

S. Urbanek and M. Theus. **iPlots**: high interaction graphics for R. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, eds. K.Hornik, F.Leisch, A.Zeileis. Vienna, March 2003. ISSN: 1609-395X, 2003.

Adrian Bowman
Department of Statistics, University of Glasgow
adrian@stats.gla.ac.uk

Ewan Crawford
Department of Statistics, University of Glasgow
ewan@stats.gla.ac.uk

Richard Bowman
Churchill College, University of Cambridge
rwb27@cam.ac.uk

R's role in the climate change debate

by Matthew Pocernich

A contentious issue within the climate research community is the validity of a sequence of reconstructions of global surface temperatures which exhibit a hockey stick-like appearance. These graphs show relatively modest hemispheric temperature variations over the past millennium followed by a sharp increase over the 20th century (See Figure 1). Climate data collected instrumentally is largely limited to the last 150 years. For information prior to this, scientists rely on proxy sources such as tree rings, ice core samples and other types of information. The reconstructions created by Mann et al. (1998) and Mann et al. (1999) have attracted particular attention due to their prominence in the last International Panel on Climate Change reports (IPCC, 2001). Various criticisms against the Mann et al. reconstructions have been voiced over the years, but most recently, articles beginning with McIntyre and McKitrick (2003) questioned whether principal component analysis (PCA) was correctly used to summarize many sources of proxy data. The contention was that the implementation of PCA using a common centering convention limited to the overlap with the instrumental data rather than using a full-length centering could potentially produce a spurious offset, even in random data. This criticism has been transformed into a much broader challenge regarding the existence of climate change.

The National Research Council (NRC) established a committee to re-evaluate Mann's work in light of the subsequent criticisms. On June 22nd, a press conference was held to announce the findings of the report and answer questions. The Chair of the committee, Dr. Gerald North, made several remarks relating sharing code and data (Board on Atmospheric Sciences and Climate, 2006).

> Our view is that all research benefits from full and open access to published datasets and that a clear explanation of analytical methods is mandatory. Peers should have access to the information needed to reproduce published results, so that increased confidence in the outcome of the study can be generated inside and outside the scientific community. Other committees and organizations have produced an extensive body of literature on the importance of open access to scientific data and on the related guidelines for data archiving and data access (e.g., NRC 1995). Paleoclimate research would benefit if individual researchers, professional societies, journal editors, and funding agencies continued to improve their efforts to ensure that these existing open access practices are followed.

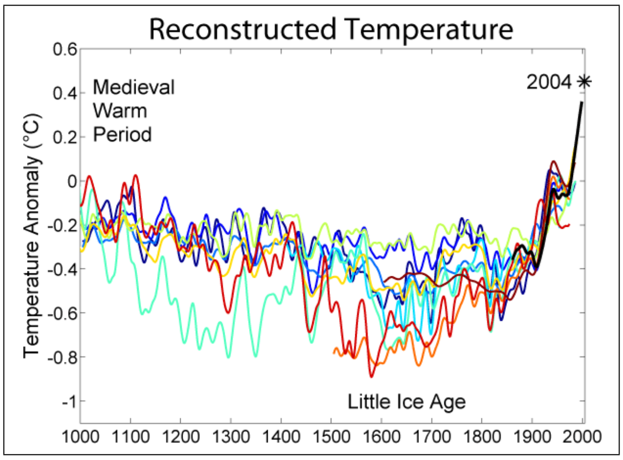


Figure 1: Reconstructed temperatures for the last millennium. Wahl and Ammann (2006)

The statistical and scientific details of this debate are not the point of this note, but rather the ways in which R has had a role in the dialog. A figure in the NRC report was created using R to illustrate how spurious trends could result from the use of principal component analysis (Figure 9-2). This is not too exciting by itself. What is more unique is that the code used to generate this figure was placed in the appendix, where it is merely referred as "R Code".

In a response to criticisms in McIntyre and McKitrick (2003), several articles were written including Wahl and Ammann (2006). Again, the issue here is not the content of the article. What is somewhat unique in the climate community is that all the code and data used to recreate the Mann's original analysis has been made available to the pubic. See www.cgd.ucar.edu/ccr/ammann/millennium/MBH_reevaluation.html. Since the analysis is in R, anyone can replicate the results and examine the methods. With such a contentious issue such as climate change, the strategy of sharing code and data may help to keep the focus on primary issue at hand: namely climate change.

R is increasingly a language of choice in the climate analysis arena. Members of the meteorological community have contributed R packages to facilitate the study climate and weather. Advanced spatial statistical tools such as thin plate spline fitting is found in **fields** (Nychka, 2005). The package **extRemes** (Gilleland et al., 2004) provides tools for applying extreme value statistics. **verification** (Pocernich, 2006) contains methods useful for verifying weather forecasts. **ensembleBMA** (Raftery et al., 2006) provides tools to create probabilistic forecasts out of an ensemble of predictions. **clim.pact** (Benestad, 2005) contains methods for downscaling data from climate models to weather models. Climate models can generate large amounts of data which is typically stored in netcdf files. The packages **ncdf** (Pierce, 2006) and **RNetCDF** (Michna, 2005) allow access to netcdf formatted files. Many of these packages have utility for those outside the weather and climate community. Undoubtedly this list is incomplete.

In discussing R and the study of climate, several points can be made. First, R code is included in the National Academy of Science report whose audience is a broad spectrum of the science and public policy community. In this report much is explained in detail or footnoted, but R was not further defined. Implicitly, this speaks to R's acceptance within the climate community. To further the discussion of climate change, in this report the NAS urges open access to statistical methods and data sharing. To this end, climate scientists such as Ammann have published their R code and data. Finally, many packages have been submitted to CRAN by the weather and climate community that both address both climatology-specific needs and more general statistical topics.

Bibliography

R. E. Benestad. *clim.pact: Climate analysis and downscaling package for monthly and daily data*. The Norwegian Meteorological Institute, 2005.

Board on Atmospheric Sciences and Climate. Surface temperature reconstructions for the last 2,000 years. Technical report, National Research Concil of the National Academies, 2006.

E. Gilleland, R. Katz, and G. Young. *extRemes: Extreme value toolkit*. National Center for Atmospheric Research, 2004. URL http://www.assessment.ucar.edu/toolkit/.

IPCC. *Climate Change 2001: The Scientific Basis*. Cambridge University Press, Cambridge, UK., 2001.

M. Mann, R. Bradley, and M. Hughes. Global-scale temperature patterns and climate forcing over the past 6 six centuries. *Nature*, 392:779–787, 1998.

M. E. Mann, R. S. Bradley, and M. K. Hughes. Northern hemisphere temperatures during the past millennium: Inferences, uncertainties, and limitations. *Geophysical Research Letters*, 26:759–762, 1999.

S. McIntyre and R. McKitrick. Corrections to the Mann et al. (1998) proxy data base and northern hemispheric average temperature series. *Energy Environment*, 14(6):751–771, 2003.

P. Michna. *RNetCDF: R Interface to NetCDF Datasets*, 2005. URL http://www.unidata.ucar.edu/packages/netcdf/. R package version 1.1-3.

D. Nychka. *fields: Tools for spatial data*. National Center for Atmospheric Research, 2005. URL http://www.image.ucar.edu/GSP/Software/Fields. R package version 2.3.

D. Pierce. *ncdf: Interface to Unidata netCDF data files*, 2006. URL http://cirrus.ucsd.edu/~pierce/ncdf. R package version 1.6.

M. Pocernich. *verification: Forecast verification utilities.* National Center for Atmospheric Research, 2006. R package version 1.08.

A. E. Raftery, J. M. Sloughter, and M. Polakowski. *ensembleBMA: Probabilistic forecasting using Bayesian Model Averaging*. University of Washington, 2006.

E. Wahl and C. Ammann. Robustness of the mann, bradley, hughes reconstruction of northern hemisphere surface temperatures: Examination of criticisms based on the nature and processing of proxy climate evidence. *Climatic Change*, in press, 2006.

Matthew Pocernich
Research Applications Laboratory
The National Center for Atmospheric Research
pocernic@rap.ucar.edu

Interacting with Data using the filehash Package

by Roger D. Peng

Overview and Motivation

Working with large datasets in R can be cumbersome because of the need to keep objects in physical memory. While many might generally see that as a feature of the system, the need to keep whole objects in memory creates challenges to those who might want to work interactively with large datasets. Here we take a simple definition of "large dataset" to be any dataset that cannot be loaded into R as a single R object because of memory limitations. For example, a very large data frame might be too large for all of the columns and rows to be loaded at once. In such a situation, one might load only a subset of the rows or columns, if that is possible.

In a key-value database, an arbitrary data object (a "value") has a "key" associated with it, usually a character string. When one requests the value associated with a particular key, it is the database's job to match up the key with the correct value and return the value to the requester.

The most straightforward example of a key-value database in R is the global environment. Every object in R has a name and a value associated with it. When you execute at the R prompt

```
> x <- 1
> print(x)
```

the first line assigns the value 1 to the name/key "x". The second line requests the value of "x" and prints out 1 to the console. R handles the task of finding the appropriate value for "x" by searching through a series of environments, including the namespaces of the packages on the search list.

In most cases, R stores the values associated with keys in memory, so that the value of x in the example above was stored in and retrieved from physical memory. However, the idea of a key-value database can be generalized beyond this particular configuration. For example, as of R 2.0.0, much of the R code for R packages is stored in a lazy-loaded database, where the values are initially stored on disk and loaded into memory on first access (Ripley, 2004). Hence, when R starts up, it uses relatively little memory, while the memory usage increases as more objects are requested. Data could also be stored on other computers (e.g. websites) and retrieved over the network.

The general S language concept of a database is described in Chapter 5 of the Green Book (Chambers, 1998) and earlier in Chambers (1991). Although the S and R languages have different semantics with respect to how variable names are looked up and bound to values, the general concept of using a key-value database applies to both languages. Duncan Temple Lang has implemented this general database framework for R in the **RObjectTables** package of Omegahat (Temple Lang, 2002). The **RObjectTables** package provides an interface for connecting R with arbitrary backend systems, allowing data values to be stored in potentially any format or location. While the package itself does not include a specific implementation, some examples are provided on the package's website.

The **filehash** package provides a full read-write implementation of a key-value database for R. The package does not depend on any external packages (beyond those provided in a standard R installation) or software systems and is written entirely in R, making it readily usable on most platforms. The **filehash** package can be thought of as a specific implementation of the database concept described in Chambers (1991), taking a slightly different approach to the problem. Both Temple Lang (2002) and Chambers (1991) focus on the notion of "attach()-ing" a database in an R/S session so that variable names can be looked up automatically via the search list. The **filehash** package represents a database as an instance of an S4 class and operates directly on the S4 object via various methods.

Key-value databases are sometimes called hash tables and indeed, the name of the package comes from the idea of having a "file-based hash table". With **filehash** the values are stored in a file on the disk rather than in memory. When a user requests the values associated with a key, **filehash** finds the object on the disk, loads the value into R and returns it to the user. The package offers two formats for storing data on the disk: The values can be stored (1) concatenated together in a single file or (2) separately as a directory of files.

Related R packages

There are other packages on CRAN designed specifically to help users work with large datasets. Two packages that come immediately to mind are the **g.data** package by David Brahm (Brahm, 2002) and the **biglm** package by Thomas Lumley. The **g.data** package takes advantage of the lazy evaluation mechanism in R via the delayedAssign function. Briefly, objects are loaded into R as promises to load the actual data associated with an object name. The first time an object is requested, the promise is evaluated and the data are loaded. From then on, the data reside in memory. The mechanism used in **g.data** is similar to the one used by the lazy-loaded databases described in Ripley (2004). The **biglm** package allows users to fit linear models on datasets that are too large to fit in memory. However, the **biglm** package does not provide methods for dealing with large datasets in general. The **filehash** package also draws inspiration from Luke Tierney's experimental **gdbm** package which implements a key-value database via the GNU dbm (GDBM) library. The use of GDBM creates an external dependence since the GDBM C library has to be compiled on each system. In addition, I encountered a problem where databases created on 32-bit machines could not be transferred to and read on 64-bit machines (and vice versa). However, with the increasing use of 64-bit machines in the future, it seems this problem will eventually go away.

The R Special Interest Group on Databases has developed a number of packages that provide an R interface to commonly used relational database management systems (RDBMS) such as MySQL (**RMySQL**), SQLite (**RSQLite**), and Oracle (**ROracle**). These packages use the classes and generics defined in the **DBI** package and have the advantage that they offer much better database functionality, inherited via the use of a true database management system. However, this benefit comes with the cost of having to install and use third-party software. While installing an RDBMS may not be an issue—many systems have them pre-installed and the **RSQLite** package comes bundled with the source for the RDBMS—the need for the RDBMS and knowledge of structured query language (SQL) nevertheless adds some overhead. This overhead may serve as an impediment for users in need of a database for simpler applications.

Creating a filehash database

Databases can be created with **filehash** using the `dbCreate` function. The one required argument is the name of the database, which we call here "mydb".

```
> library(filehash)
```

```
Simple key-value database (version
0.8-1 2006-09-25)
```

```
> dbCreate("mydb")
```

```
[1] TRUE
```

```
> db <- dbInit("mydb")
```

You can also specify the `type` argument which controls how the database is represented on the backend. We will discuss the different backends in further detail later. For now, we use the default backend which is called "DB1".

Once the database is created, it must be initialized in order to be accessed. The `dbInit` function returns an S4 object inheriting from class "filehash". Since this is a newly created database, there are no objects in it.

Accessing a filehash database

The primary interface to filehash databases consists of the functions `dbFetch`, `dbInsert`, `dbExists`, `dbList`, and `dbDelete`. These functions are all generic—specific methods exists for each type of database backend. They all take as their first argument an object of class "filehash". To insert some data into the database we can simply call `dbInsert`

```
> dbInsert(db, "a", rnorm(100))
```

```
[1] TRUE
```

Here we have associated with the key "a" 100 standard normal random variates. We can retrieve those values with `dbFetch`.

```
> value <- dbFetch(db, "a")
> mean(value)
```

```
[1] 0.002912563
```

The function `dbList` lists all of the keys that are available in the database, `dbExists` tests to see if a given key is in the database, and `dbDelete` deletes a key-value pair from the database

```
> dbInsert(db, "b", 123)
```

```
[1] TRUE
```

```
> dbDelete(db, "a")
```

```
[1] TRUE
```

```
> dbList(db)
```

```
[1] "b"
```

```
> dbExists(db, "a")
```

```
[1] FALSE
```

While using functions like `dbInsert` and `dbFetch` is straightforward it can often be easier on the fingers to use standard R subset and accessor functions like `$`, `[[`, and `[`. Filehash databases have methods for these functions so that objects can be accessed in a more compact manner. Similarly, replacement methods for these functions are also available. The `[` function can be used to access multiple objects from the database, in which case a list is returned.

```
> db$a <- rnorm(100, 1)
> mean(db$a)
```

```
[1] 1.011141

> mean(db[["a"]])

[1] 1.011141

> db$b <- rnorm(100, 2)
> dbList(db)

[1] "a" "b"
```

For all of the accessor functions, only character indices are allowed. Numeric indices are caught and an error is given.

```
> e <- local({
+     err <- function(e) e
+     tryCatch(db[[1]], error = err)
+ })
> conditionMessage(e)

[1] "numeric indices not allowed"
```

Finally, there is method for the `with` generic function which operates much like using `with` on lists or environments.

The following three statements all return the same value.

```
> with(db, c(a = mean(a), b = mean(b)))

       a        b
1.011141 2.012793
```

When using `with`, the values of “a” and “b” are looked up in the database.

```
> sapply(db[c("a", "b")], mean)

       a        b
1.011141 2.012793
```

Here, using `[` on `db` returns a list with the values associated with “a” and “b”. Then `sapply` is applied in the usual way on the returned list.

```
> unlist(lapply(db, mean))

       a        b
1.011141 2.012793
```

In the last statement we call `lapply` directly on the “filehash” object. The **filehash** package defines a method for `lapply` that allows the user to apply a function on all the elements of a database directly. The method essentially loops through all the keys in the database, loads each object separately and applies the supplied function to each object. `lapply` returns a named list with each element being the result of applying the supplied function to an object in the database. There is an argument `keep.names` to the `lapply` method which, if set to `FALSE`, will drop all the names from the list.

Loading filehash databases

An alternative way of working with a filehash database is to load it into an environment and access the element names directly, without having to use any of the accessor functions. The **filehash** function `dbLoad` works much like the standard R `load` function except that `dbLoad` loads active bindings into a given environment rather than the actual data. The active bindings are created via the `makeActiveBinding` function in the **base** package. `dbLoad` takes a filehash database and creates symbols in an environment corresponding to the keys in the database. It then calls `makeActiveBinding` to associate with each key a function which loads the data associated with a given key. Conceptually, active bindings are like pointers to the database. After calling `dbLoad`, anytime an object with an active binding is accessed the associated function (installed by `makeActiveBinding`) loads the data from the database.

We can create a simple database to demonstrate the active binding mechanism.

```
> dbCreate("testDB")

[1] TRUE

> db <- dbInit("testDB")
> db$x <- rnorm(100)
> db$y <- runif(100)
> db$a <- letters
> dbLoad(db)
> ls()

[1] "a"  "db" "x"  "y"
```

Notice that we appear to have some additional objects in our workspace. However, the values of these objects are not stored in memory—they are stored in the database. When one of the objects is accessed, the value is automatically loaded from the database.

```
> mean(y)

[1] 0.5118129

> sort(a)

 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i"
[10] "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z"
```

If I assign a different value to one of these objects, its associated value is updated in the database via the active binding mechanism.

```
> y <- rnorm(100, 2)
> mean(y)

[1] 2.010489
```

If I subsequently remove the database and reload it later, the updated value for "y" persists.

```
> rm(list = ls())
> db <- dbInit("testDB")
> dbLoad(db)
> ls()

[1] "a"  "db" "x"  "y"

> mean(y)

[1] 2.010489
```

Perhaps one disadvantage of the active binding approach taken here is that whenever an object is accessed, the data must be reloaded into R. This behavior is distinctly different from the the delayed assignment approach taken in **g.data** where an object must only be loaded once and then is subsequently in memory. However, when using delayed assignments, if one cycles through all of the objects in the database, one could eventually exhaust the available memory.

Cleaning up

When a database is initialized using the default "DB1" format, a file connection is opened for reading and writing to the database file on the disk. This file connection remains open until the database is closed via `dbDisconnect` or the database object in R is removed. Since there is a hard limit on the number of file connections that can be open at once, some protection is need to make sure that file connections are close properly.

Upon initialization, the database stores the file connection number in an environment and registers a finalizer to close the connection. Once a database is closed or removed and garbage collection occurs, the finalizer is called and the file connection closed appropriately.

A larger example

As an example of using **filehash** for interacting with a much larger database, we will use some air pollution and weather data that were originally collected as part of the National Morbidity, Mortality, and Air Pollution Study (Samet et al., 2000a,b). The full database, which includes daily mortality data, is available from the **NMMAPSdata** package (Peng and Welty, 2004). For the purposes of this example, we have extracted just the air pollution and weather data.

Briefly, the database consists of data from 108 cities in the United States. For each city, there is a data frame of 5114 rows and 274 columns. The database contains daily measurements for the U.S. Environmental Protection Agency's "criteria" pollutants—particulate matter, nitrogen dioxide, sulphur dioxide, ozone, carbon monoxide—as well as temperature, dew point temperature, and relative humidity. For the gaseous pollutants, hourly values are available. For particulate matter (both PM_{10} and $PM_{2.5}$), 24 hour averages are available. The filehash database can be downloaded from http://www.biostat.jhsph.edu/~rpeng/RR/.

Here, we load the database and list the first 10 cities.

```
> db <- dbInit("metpoll")
> dbLoad(db)
> ls()[1:10]

 [1] "akr"  "albu" "anch" "arlv" "atla"
 [6] "aust" "bake" "balt" "batr" "bidd"
```

The keys for the database are abbreviated versions of the city names. The full city names can be found in the "citynames.csv" data frame available from the author's website.

Normally, it would have been impossible to load all of the data from all of the cities at once, but `dbLoad` merely creates active bindings for the data. As our session progresses, data for each city will be dynamically loaded as needed.

Now we can simply access each city's data frame by its abbreviated name, as if every data frame had already been loaded into the workspace. For example we can regress daily PM_{10} on temperature in Chicago, Illinois,

```
> lm(pm10tmean ~ tmpd, data = chic)

Call:
lm(formula = pm10tmean ~ tmpd, data = chic)

Coefficients:
(Intercept)         tmpd
   -18.0002       0.3565
```

or in New York City, New York.

```
> lm(pm10tmean ~ tmpd, data = ny)

Call:
lm(formula = pm10tmean ~ tmpd, data = ny)

Coefficients:
(Intercept)         tmpd
   -17.3641       0.3121
```

Meanwhile, our overall memory usage has only increased slightly since initially starting up R.

```
> gc()[, 1:2]

         used (Mb)
Ncells 303654  8.2
Vcells 142479  1.1
```

The **filehash** package is most useful in situations where one does not require simultaneous access to all the data at once. In that situation, one would have to have all the data loaded into memory in order to operate on it (e.g. fit models, compute statistics) and **filehash** would not be able to help. However, in situations where one would like to have ready access to the data and can operate on single elements or perhaps small groups of elements at a time, the **filehash** package may be of use.

Other filehash utilities

There are a few other utilities included with the **filehash** package. Two of the utilities, `dumpObjects` and `dumpImage`, are analogues of `save` and `save.image`. Rather than save objects to an R workspace, `dumpObjects` saves the given objects to a "filehash" database so that in the future, individual objects can be reloaded if desired. Similarly, `dumpImage` saves the entire workspace to a "filehash" database.

The function `dumpList` takes a list and creates a "filehash" database with values from the list. The list must have a non-empty name for every element in order for `dumpList` to succeed. `dumpDF` creates a "filehash" database from a data frame where each column of the data frame is an element in the database. Essentially, `dumpDF` converts the data frame to a list and calls `dumpList`.

Filehash database backends

Currently, the **filehash** package can represent databases in two different formats. The default format is called "DB1" and it stores the keys and values in a single file. From experience, this format works well overall but can be a little slow to initialize when there are many thousands of keys. Briefly, the "filehash" object in R stores a map which associates keys with a byte location in the database file where the corresponding value is stored. Given the byte location, we can `seek` to that location in the file and read the data directly. Before reading in the data, a check is made to make sure that the map is up to date. This format depends critically on having a working `ftell` at the system level and a crude check is made when trying to initialize a database of this format.

The second format is called "RDS" and it stores objects as separate files on the disk in a directory with the same name as the database. This format is the most straightforward and simple of the available formats. When a request is made for a specific key, **filehash** finds the appropriate file in the directory and reads the file into R. The only catch is that on operating systems that use case-insensitive file names, objects whose names differ only in case will collide on the filesystem. To workaround this, object names with capital letters are stored with mangled names on the disk. An advantage of this format is that most of the organizational work is delegated to the filesystem.

There is a third format called "DB" and it is a predecessor of the "DB1" format. This format is like the "DB1" format except the map which associates keys to byte locations is stored in a separate file. Therefore, each database is represented by two separate files—an index file and a data file. This format is retained for back compatibility but users should generally try to use the "DB1" format instead.

Extending filehash

The **filehash** package has a mechanism for developing new backend formats, should the need arise. The function `registerFormatDB` can be used to make **filehash** aware of a new database format that may be implemented in a separate R package or a file. `registerFormatDB` takes two arguments: a `name` for the new format (like "DB1" or "RDS") and a list of functions. The list should contain two functions: one function named "create" for creating a database, given the database name, and another function named "initialize" for initializing the database. In addition, one needs to define methods for `dbInsert`, `dbFetch`, etc.

A list of available backend formats can be obtained via the `filehashFormats` function. Upon registering a new backend format, the new format will be listed when `filehashFormats` is called.

The interface for registering new backend formats is still experimental and could change in the future.

Discussion

The **filehash** package has been designed be useful in both a programming setting and an interactive setting. Its main purpose is to allow for simpler interaction with large datasets where simultaneous access to the full dataset is not needed. While the package may not be optimal for all settings, one goal was to write a simple package in pure R that users to could install with minimal overhead. In the future I hope to add functionality for interacting with databases stored on remote computers and perhaps incorporate a "real" database backend. Some work has already begun on developing a backend based on the **RSQLite** package.

Bibliography

D. E. Brahm. Delayed data packages. *R News*, 2(3):11–12, December 2002. URL `http://CRAN.R-project.org/doc/Rnews/`.

J. M. Chambers. Data management in S. Technical Report 99, AT&T Bell Laboratories Statistics Research, December 1991. http://stat.bell-labs.com/doc/93.15.ps.

J. M. Chambers. *Programming with Data: A Guide to the S Language*. Springer, 1998.

R. D. Peng and L. J. Welty. The NMMAPSdata package. *R News*, 4(2):10–14, September 2004. URL http://CRAN.R-project.org/doc/Rnews/.

B. D. Ripley. Lazy loading and packages in R 2.0.0. *R News*, 4(2):2–4, September 2004.

J. M. Samet, F. Dominici, S. L. Zeger, J. Schwartz, and D. W. Dockery. *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodological Issues*. Health Effects Institute, Cambridge MA, 2000a.

J. M. Samet, S. L. Zeger, F. Dominici, F. Curriero, I. Coursac, D. W. Dockery, J. Schwartz, and A. Zanobetti. *The National Morbidity, Mortality, and Air Pollution Study, Part II: Morbidity and Mortality from Air Pollution in the United States*. Health Effects Institute, Cambridge, MA., 2000b.

D. Temple Lang. *RObjectTables: User-level attach()'able table support*, 2002. URL http://www.omegahat.org/RObjectTables. R package version 0.3-1.

Roger D. Peng
Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health
rpeng@jhsph.edu
http://www.biostat.jhsph.edu/~rpeng/

Special functions in R: introducing the gsl package

An R wrapper for the Gnu Scientific Library

by Robin K. S. Hankin

Introduction

The Gnu Scientific Library (GSL) is a collection of numerical routines for scientific computing (Galassi et al., 2005). The routines are written in C and constitute a library for C programmers; the source code is distributed under the GNU General Program License. One stated aim of the GSL development effort is the development of wrappers for high level languages.

The R programming language (R Development Core Team, 2005) is an environment for statistical computation and graphics. It consists of a language and a run-time environment with graphics and other features.

Here I introduce **gsl**, an R package that allows direct access to many GSL functions, including all the special functions, from within an R session. The package is available on CRAN, at `http://www.cran.r-project.org/`; the GSL is available at `http://www.gnu.org/software/gsl/`.

Package design philosophy

The package splits into two parts: the special functions, written by the author; and the **rng** and **qrng** functionality, written by Duncan Murdoch. These two parts are very different in implementation, yet follow a common desideratum, namely that the package be a transparent port of the GSL library. The package thus has the advantage of being easy to compare with the GSL, and easy to update verifiably.

In this paper, the Airy functions are used to illustrate the package. They are typical of the package's capabilities and coding, and are relatively simple to understand, having only a single real argument.

The package is organized into units that correspond to the GSL header file. Thus all the Airy functions are defined in a single header file, `gsl_sf_airy.h`. The package thus contains a corresponding C file, `airy.c`; an R file, `airy.R`, and a documentation file, `Airy.Rd`. These three files together encapsulate the functionality defined in `gsl_sf_airy.h` in the context of an R package. This structure makes it demonstrable the the GSL has been systematically and completely wrapped.

Functions are named such that one can identify a function in the GSL manual, and the corresponding R command will be the same but with the prefix[1] and, if present, the "`_e`" suffix, removed. In the case of the special functions, the prefix is "`gsl_sf_`". Thus, GSL function `gsl_sf_airy_Ai_e()` of header file `gsl_sf_airy.h` is called, via intermediate C routine `airy_Ai_e()`, by R function `airy_Ai()`. Documentation is provided for every function defined in `gsl_sf_airy.h` under `Airy.Rd`.

The **gsl** package is not intended to add any nu-

[1] Some functions, such as `gsl_sf_sin()`, retain the prefix to avoid conflicts. A full list is given in `Misc.Rd`.

merical functionality to the GSL, although here and there I have implemented slight extensions such as the Jacobian elliptic functions whose R ports take a complex argument.

Package documentation

The **gsl** package is unusual in that its documentation consists almost entirely of pointers to the GSL reference manual (Galassi et al., 2005), and to Abramowitz and Stegun (1965). This follows from the transparent wrapper philosophy. In any case, the GSL reference manual would strictly dominate the `Rd` files of the **gsl** package.

Package gsl in use

Most functions in the package are straightforwardly and transparently executable:

```
> airy_Ai(1:3)
[1] 0.13529242 0.03492413 0.00659114
```

The online helpfiles include many examples that reproduce graphs and tables that appear in Abramowitz and Stegun (1965). This constitutes a useful check on the routines. For example, Figure 1 shows an approximate reproduction of their Figure 10.6 (page 446).

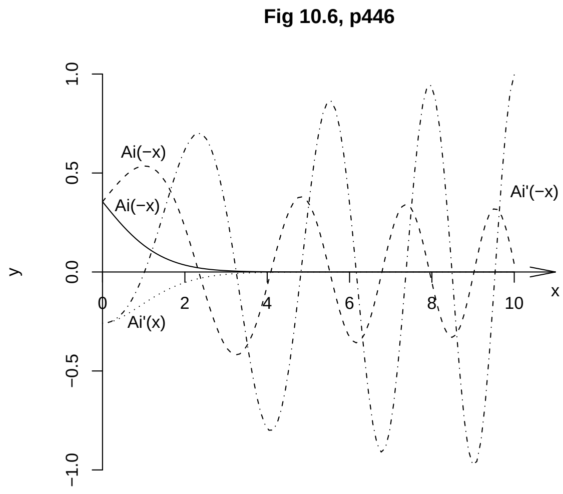


Figure 1: Functions Ai$(\pm x)$ and Ai$'(\pm x)$ as plotted in the helpfile for `airy_Ai()` and appearing on page 446 of Abramowitz and Stegun (1965)

Summary

The **gsl** package is a transparent R wrapper for the Gnu Scientific Library. It gives access to all the special functions and the quasi-random sequence generation routines. Notation follows the GSL as closely as reasonably practicable; many graphs and tables appearing in Abramowitz and Stegun (1965) are reproduced by the examples in the helpfiles.

Acknowledgments

I would like to acknowledge the many stimulating and helpful comments made by the R-help list over the years.

Bibliography

M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions*. New York: Dover, 1965.

M. Galassi et al. *GNU Scientific Library*, 2005. Reference Manual edition 1.7, for GSL version 1.7; 13 September 2005.

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Robin K. S. Hankin
National Oceanography Centre, Southampton
European Way
Southampton
United Kingdom
SO14 3ZH
`r.hankin@noc.soton.ac.uk`

A short Introduction to the SIMEX and MCSIMEX

by Wolfgang Lederer and Helmut Küchenhoff

In statistical practice variables are often contaminated with measurement error. This may be the case due to bad measurement tools or because the true variable can not be measured directly. In the case of discrete variables, measurement error is referred to as misclassification. In the framework of general regression, measurement error or misclassification can lead to serious bias in the estimated parameters. In most cases the estimated effect of the contaminated variable is attenuated, e.g., see Carroll et al. (2006).

Among other methods the simulation and extrapolation method (SIMEX) by Cook and Stefanski (1994) has become a useful tool for correcting effect estimates in the presences of additive measurement error. The method is especially helpful for complex models with a simple measurement error structure. The same basic idea of simulation and extrapolation has been transferred to the case of misclassification (MC–SIMEX) by Küchenhoff et al. (2006).

The R package **simex** provides functions to use the SIMEX or MC–SIMEX methods for various kinds of regression objects and to produce graphics and summary statistics for corrected objects. There are also functions to provide help in constructing misclassification matrices.

Theory

We want to estimate an effect parameter (vector) β in a general regression model in the presence of misclassification or measurement error. We assume that an estimator, which is consistent when all variables are measured without error, is available. This estimator is called the naive estimator when it is used although there is measurement error in the data. The SIMEX–method uses the relationship between the size of the measurement error, described by the measurement error variance σ_u^2 and the bias of the effect estimator when ignoring the measurement error. So we can define the function

$$\sigma_u^2 \longrightarrow \beta^*(\sigma_u^2) := \mathcal{G}(\sigma_u^2)$$

where β^* is the limit to which the naive estimator converges as the sample size $n \rightarrow \infty$. Consistency implies that $\mathcal{G}(0) = \beta$. The idea of the SIMEX method is to approximate the function $\mathcal{G}(\sigma_u^2)$ by a parametric approach $\mathcal{G}(\sigma^2, \Gamma)$, for example with a quadratic approximation $\mathcal{G}_{quad}(\sigma_u^2, \Gamma) = \gamma_0 + \gamma_1\sigma_u^2 + \gamma_2(\sigma_u^2)^2$.

In the simulation step, to estimate Γ, the method adds measurement error with variance $\lambda\sigma_u^2$ to the contaminated variable, where $\lambda > 0$ quantifies the additional amount of measurement error that is added. The resulting measurement error variance is then $(1+\lambda)\sigma_u^2$. The naive estimator for this increased measurement error is calculated. This simulation procedure is repeated B times. The average over the B estimators estimates $\mathcal{G}((1+\lambda)\sigma_u^2)$. Performing these simulations for a fixed grid of λs, leads to an estimator for $\hat{\Gamma}$ of the parameters $\mathcal{G}(\sigma_u^2, \Gamma)$, for example by least squares. Simulation results indicate that $\lambda \in (0.5, 1, 1.5, 2)$ is a good choice in most cases.

In the extrapolation step the approximated function $\mathcal{G}(\sigma_u^2, \hat{\Gamma})$ is extrapolated back to the case of no measurement error and so the SIMEX estimator is defined by $\hat{\beta}_{simex} := \mathcal{G}(0, \hat{\Gamma})$, which corresponds to $\lambda = -1$.

The misclassification error can be described by the misclassification matrix Π which is defined via its components

$$\pi_{ij} = P(X^* = i | X = j)$$

where X^* is the misclassified version of X. Π is a $k \times k$ matrix where k is the number of possible outcomes of X. The estimator β^* depends on the amount of misclassification and is defined by

$$\lambda \longrightarrow \beta^*\left(\Pi^\lambda\right)$$

where Π^λ is defined via its spectral decomposition $\Pi^\lambda := E\Lambda^\lambda E^{-1}$, with Λ being the diagonal matrix of eigenvalues and E the corresponding matrix of eigenvectors. This allows the SIMEX method to be applied to misclassification problems. The MC-SIMEX estimator is then defined by the parametric approximation of the function

$$\lambda \rightarrow \beta^*\left(\Pi^\lambda\right) \approx \mathcal{G}_\Pi(1+\lambda, \Gamma).$$

In the simulation step we simulate B new pseudo data sets for a fixed grid of λ by the misclassification operation defined by

$$X_i^* := \text{MC}\left[\Pi^\lambda\right](X_i).$$

The misclassification operation $\text{MC}[M](X_i)$ generates by simulation a misclassified version of the true, but unknown, variable X_i denoted by X_i^* which is related to X_i by the misclassification matrix M. For each of these B pseudo data sets the naive estimators are calculated and averaged for each λ. These averaged naive estimators converge to $\mathcal{G}_\Pi(1+\lambda, \Gamma)$ and the estimation of Γ via, e.g., least squares and so an approximation of $\mathcal{G}_\Pi(1+\lambda, \Gamma)$ is feasible.

The MCSIMEX estimator is then defined by

$$\beta_{MCSIMEX} := \mathcal{G}(0, \hat{\Gamma})$$

which corresponds again to $\lambda = -1$. One requirement for the application of the (MC-)SIMEX Method is that the measurement error variance or the misclassification matrix, respectively, has to be known or can be estimated by additional validation information.

Variance estimation

The ease of getting corrected parameter estimates is somewhat offset by the complexity of the calculation of the parameter's standard error. With its simulation character it is a natural candidate for the bootstrap. Although this is a valid method for obtaining standard errors, it is rather time consuming and for complex models not feasible. We implemented two other methods for the estimation of standard errors which have a smaller computational burden. The jackknife method was developed for the SIMEX method by Stefanski and Cook (1995). For the MC-SIMEX method, it lacks theoretical foundation but simulation results indicate valid estimates for the standard errors.

An asymptotic approach based on estimation equations was developed by Carroll et al. (1996) for the SIMEX method and extended to the MCSIMEX method by Küchenhoff et al. (2006). It is possible to take the uncertainty of an estimated misclassification matrix or an estimated measurement error variance into account.

Example

To illustrate the application of the simex package, we use a data set of a study about chronic bronchitis and dust concentration of the Deutsche Forschungsgemeinschaft (German research foundation). The data were recorded during the years 1960 and 1977 in a Munich plant (1246 workers). The data can be downloaded from `http://www.stat.uni-muenchen.de/service/datenarchiv/dust/dust_e.html`.

The data set contains 4 variables described in Table 1, and is read into the `data.frame` *dat* via the `read.table` command.

The naive model is then given by

```
> naive <- glm(cbr ~ dust + smoking + expo,
+          family= binomial,
+          data =dat, x=TRUE, y=TRUE)
```

where *cbr* and *smoking* must be `factors`. The options 'x','y' save the information about the response and the variables in the glm object and must be enabled for asymptotic variance estimation.

Continuous data

It is possible, that the variable *dust* is subject to measurement error. Because it is a continuous variable the SIMEX-method is to be used here. Usually the measurement error variance is estimated by replication or by a validation study, see Carroll et al. (2006), which is not available in this case. For illustration we assume that the additive measurement error has a standard deviation $\sigma = 2$. The estimation by the `simex` function is done as follows.

```
> mod.sim <- simex(naive,
+             measurement.error = 2,
+             SIMEXvariable="dust",
+             fitting.method ="quad")
> mod.sim

Naive model:
glm(formula = cbr ~ dust + smoking + expo,
             family = binomial,
             data = dat, x = TRUE, y = TRUE)

SIMEX-Variables: dust
Number of Simulations: 100

Coefficients:
(Intercept)      dust   smoking1      expo
   -3.16698   0.13549    0.67842   0.03969
```

The default extrapolation function ("fitting.method") for the function `simex` is a quadratic polynomial, which has a good performance in many cases. Other possibilities are a linear, a loglinear and a nonlinear extrapolation function. Unfortunately, the nonlinear extrapolation is numerically not stable and it is therefore advised to use it via the `refit` function. The `refit` function fits a new extrapolation function to the data obtained by the simulation step and yields therefore different estimators. It can be applied to objects of class MCSIMEX as well.

```
> refit(mod.sim, "nonl")

Naive model:
glm(formula = cbr ~ dust + smoking + expo,
           family = binomial,
           data = dat, x = TRUE, y = TRUE)

SIMEX-Variables: dust
Number of Simulations: 100

Coefficients:
(Intercept)      dust   smoking1      expo
   -3.33167   0.18904    0.67854   0.03956

> coef(naive)
(Intercept)      dust     smoking1      expo
   -3.04787   0.09189     0.67684   0.04015
```

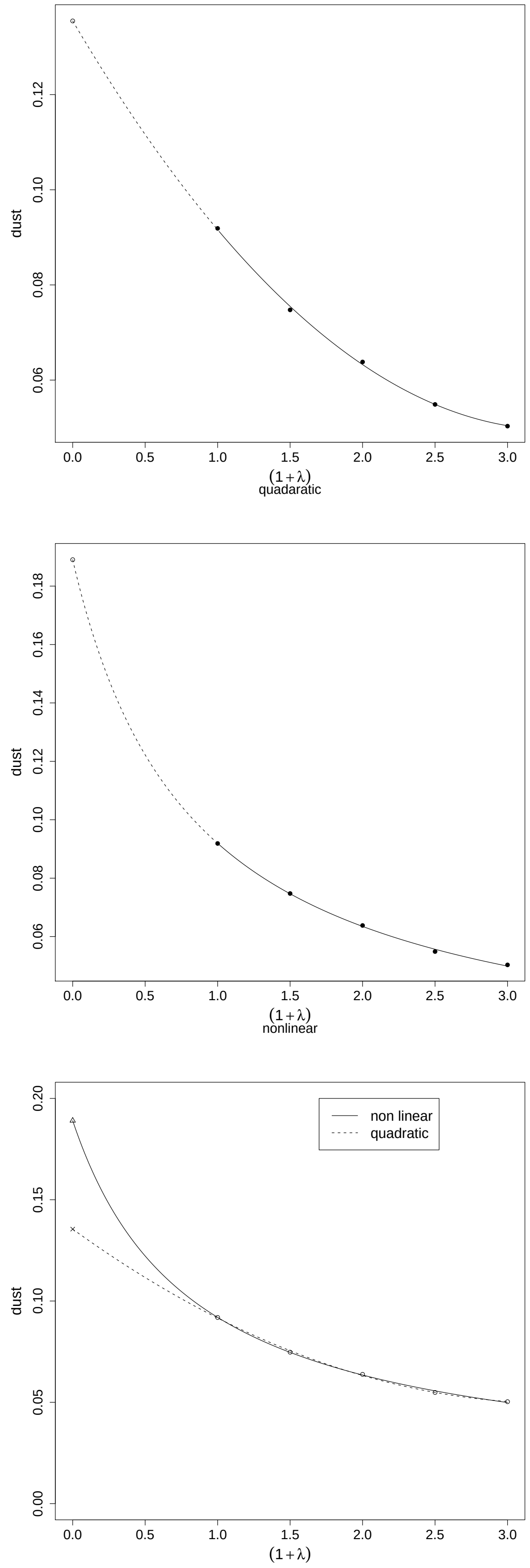


Figure 1: The effects of measurement error in variable *dust* produced with `plot(mod.sim,ask = c(FALSE,TRUE,FALSE,FALSE))` and `plot(refit(mod.sim,"nonl"),ask = c(FALSE,TRUE,FALSE,FALSE))` and combined in one diagram. Note that the point relating to $1+\lambda = 1$ corresponds to the naive estimate.

cbr	Chronic Bronchitis Reaction	1 : Yes 0 : No
dust	Dust concentration at work	(in mg / m^3)
smoking	Does worker smoke?	1 : Yes 0 : No
expo	Duration of exposure	in years

Table 1: Description of variables in the bronchitis data set.

As can be seen from the displayed naive estimates, there is nearly no measurement error correction for the estimates of the *Intercept*, *smoking* and *expo*, while the corrected estimate for the variable *dust* differs substantially form the naive estimator. A comparison of both extrapolation functions for the last estimator is shown in Figure 1. Since the theory does not provide an optimal choice for the extrapolation function, different choices should be calculated and inspected graphically.

Discrete Data

It is known that some participants do not tell the truth, when asked about their smoking behavior. Research from other studies indicates, that about 8% of smokers self-report them as non-smokers. So we use the misclassification matrix for smoking defined by

```
> mc.s <- matrix(c(1,0,0.08,0.92),nrow=2)
> dimnames(mc.s) <- list(levels(dat$smoking),
+                       levels(dat$smoking))
> mc.s
  0    1
0 1 0.08
1 0 0.92
```

and so the MCSIMEX-Algorithm can be used by calling the function `mcsimex()` and a quick overview can be obtained using the `print` method.

```
>mod.smoking <- mcsimex(naive,
+                 mc.matrix = mc.s,
+                 SIMEXvariable = "smoking")
> mod.smoking

Naive model:
glm(formula = cbr ~ dust + smoking + expo,
      family = binomial, data = dat,
      x = TRUE, y = TRUE)

SIMEX-Variables: smoking
Number of Simulations: 100

Coefficients:
(Intercept)       dust    smoking1       expo
   -3.25827    0.09269     0.88086    0.04026
```

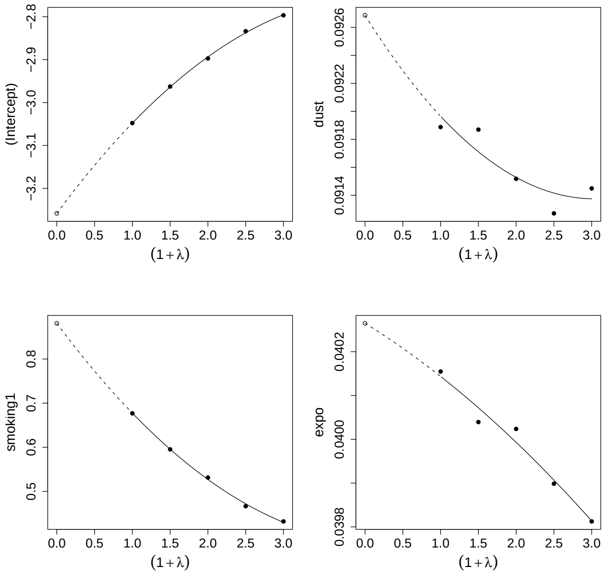


Figure 2: The effects of misclassification of smoking produced with `plot(mod.smoking,mfrow = c(2,2))`.

More detailed information is available through the `summary` method, as shown in Figure 3 and the `plot` method which is shown in Figure 2. Notice that the correction of the effect estimates *intercept, dust* and *expo* are rather small in absolute value, but that *smoking* is corrected rather strongly.

Misclassification might as well occur in the response *cbr*. It is possible to correct for just the response or for both. In the following the code for correction of the response *cbr* and the variable *smoking* is shown. For illustration we assume that the misclassification probabilities are:
$P(CBR_{Obervered} = 1|CBR_{true} = 0) = 0.2$ and
$P(CBR_{Obervered} = 0|CBR_{true} = 1) = 0.1$.

```
> mc.cbr <- matrix(c(0.8,0.2,0.1,0.9),nrow=2)
> dimnames(mc.cbr) <- list(levels(dat$cbr),
+                          levels(dat$cbr))
> mc.cbr
    0   1
0 0.8 0.1
1 0.2 0.9
> mod.both <- mcsimex(naive,
+   mc.matrix =
+     list(smoking = mc.s, cbr = mc.cbr),
+   SIMEXvariable = c("cbr", "smoking"))
> mod.both
> summary(mod.smoking)

Call: mcsimex(model = naive, SIMEXvariable = "smoking", mc.matrix = mc.s)

Naive model:
glm(formula = cbr ˜ dust + smoking + expo, family = binomial,
    data = dat, x = TRUE, y = TRUE)

Simex variable : smoking
Misclassification matrix:
  0    1
0 1 0.08
1 0 0.92

Number of iterations:  100

Residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.610900 -0.259600 -0.149800  0.006416 -0.057690  0.941900

Coefficients:

Asymptotic variance:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.258272   0.286575 -11.370  < 2e-16 ***
dust         0.092687   0.023967   3.867 0.000116 ***
smoking1     0.880861   0.240993   3.655 0.000268 ***
expo         0.040265   0.005942   6.777 1.89e-11 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Jackknife variance:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.258272   0.280754 -11.605  < 2e-16 ***
dust         0.092687   0.023366   3.967 7.70e-05 ***
smoking1     0.880861   0.217463   4.051 5.42e-05 ***
expo         0.040265   0.006251   6.442 1.69e-10 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

Figure 3: Output of `summary(mod.smoking)`.

```
Naive model:
glm(formula = cbr ˜ dust + smoking + expo,
     family = binomial,
     data = dat, x = TRUE, y = TRUE)

SIMEX-Variables: cbr, smoking
Number of Simulations: 100

Coefficients:
(Intercept)      dust   smoking1     expo
   -5.58519   0.16081    1.36969  0.07154
```

It is possible to model more complex kinds of misclassification, e.g., dependent misclassification, by submitting the name of a function that returns the misclassified variables, instead of a misclassification matrix to the function `mcsimex`. Although the SIMEX method could also be applied to situations where one variable is misclassified and another variable has additive measurement error, this is not implemented in our package.

Summary

The package **simex** features easy to use functions for correcting estimation in regression models with measurement error or misclassification via the SIMEX– or MCSIMEX–method. It provides fast and easy means to produce plots that illustrate the effect of measurement error or misclassification on parameters. Several additional functions are available that help with various problems concerning misclassification or measurement error.

Bibliography

R. J. Carroll, D. Ruppert, L. A. Stefanski and C. Crainiceanu. Measurement Error in Nonlinear Models: A Modern Perspektive, Second Edition. Chapman and Hall, CRC press, 2006.

R. J. Carroll, H. Küchenhoff, F. Lombard, and L. A. Stefanski. Asymptotics for the SIMEX Estimator in Nonlinear Measurement Error Models. *Journal of the American Statistical Association*, 91: 242–250, 1996.

J. R. Cook and L. A. Stefanski. Simulation–Extrapolation Estimation in Parametric Measurement Error Models. *Journal of the American Statistical Association*, 89:1314–1328, 1994.

H. Küchenhoff, W. Lederer, and E. Lesaffre. Asymptotic Variance Estimation for the Misclassification SIMEX, 2006. Discussion paper 473, SFB 386, Ludwig Maximilians Universität München, www.stat.uni-muenchen.de/sfb386/.

H. Küchenhoff, S. M. Mwalili, and E. Lesaffre. A general method for dealing with misclassification in regression: the Misclassification SIMEX. *Biometrics*, 62:85–96, 2006.

L. A. Stefanski and J. R. Cook. Simulation-Extrapolation: The Measurement Error Jackknife. *Journal of the American Statistical Association*, 90: 1247–1256, 1995.

Wolfgang Lederer
Ludwig–Maximilians–Universität München
Statistical Consulting Unit
Wolfgang.Lederer@googlemail.com

Helmut Küchenhoff
Ludwig–Maximilians–Universität München
Statistical Consulting Unit
Kuechenhoff@stat.uni-muenchen.de

Parametric Links for Binary Response

by Roger Koenker

It is a textbook cliché that probit and logit link functions for binary response models are "very similar." While true, this remark is unfortunately often extrapolated to imply that *all* links are essentially equivalent. Like most extrapolations, this inferential leap is dangerous. Some relatively minor recent changes in the `glm` function of R 2.4.0 make it easy to explore non-traditional link functions. This note is intended to illustrate this functionality, and encourage others to explore alternative links.

Since the early eighties, I've been teaching applied econometrics for economics graduate students at the University of Illinois. When we come to binary response models I always introduce the Pregibon (1980) "goodness of link" test, partly as a way to reinforce the ideas underlying the Box-Cox transformation and partly because I feel it is important to convey a more skeptical attitude about the conventional choice of logit and probit links.

Pregibon considers a two parameter generalization of the logit link that we can write as,

$$g(u, a, b) = \frac{u^{a-b} - 1}{a - b} - \frac{(1-u)^{a+b} - 1}{a + b}.$$

When $a = b = 0$ the link reduces to the logit expression $\log(u/(1-u))$, for $b = 0$ we have a family of symmetric densities with a controlling the heaviness of the tails, and when $b \neq 0$ it controls the skewness of the distribution. The function g is the quantile function of a version of the generalized Tukey-λ distributions, for which the **gld** package of King (2005) provides "p-q-r-d" functions for R.

Another natural (one-parameter) family of link functions for binary response is the Gosset, or Student-t, link with a free degrees of freedom parameter, ν. When $\nu = 1$ we have the so-called Cauchit link corresponding to the Cauchy quantile function, while for $\nu \to \infty$ we obtain the probit link. The Cauchit link *is* distinctly different than probit – admitting that even extreme values of the linear predictor can occasionally be overwhelmed by an even more extreme realization of the Cauchy innovation of the latent variable form of the binary response model. See Morgan and Smith (1992) for an example of Cauchit fitting.

Implementation

Parametric links may be specified in R as a structure consisting of functions specifying the link function, the inverse of the link function, the derivative of the inverse link function, a validity check function, and a name to be used for the link. This object should be of class `link-glm`. The Gosset link is:

```
Gosset <- function(nu) {
  qqt <- function(p, nu)
     sign(p-0.5)*sqrt(qf(1-2*pmin(p,1-p), 1, nu))
  linkfun <- function(mu) qqt(mu,nu)
  linkinv <- function(eta) {
    thresh <- -qqt(.Machine$double.eps,nu)
    eta <- pmin(thresh, pmax(eta, -thresh))
    pt(eta, nu)
   }
   mu.eta <- function(eta)
     pmax(dt(eta, nu), .Machine$double.eps)
   valideta <- function(eta) TRUE
   name <- "Gosset"
   structure(list(linkfun=linkfun,
     linkinv=linkinv,
     mu.eta=mu.eta,
     valideta=valideta,
     name=name),
     class = "link-glm")
}
```

Note that qt has been replaced by qqt since the former has a restricted domain for $\nu \geq 1$ while the version based on qf is reliable for $\nu \geq 0.2$. (Thanks to Luke Tierney for this suggestion.)

The Pregibon link is implemented like this:

```
Pregibon <- function(a, b) {
  linkfun <- function(mu)
    - qPregibon(1 - mu,a = a, b = b)
  linkinv <- function(eta) {
    eps <- .Machine$double.eps^.5
    tlo <- qPregibon(eps, a = a, b = b)
    thi <- qPregibon(1 - eps, a = a, b = b)
    eta <- -pmin(thi, pmax(-eta, tlo))
    1 - pPregibon(-eta, a = a, b = b)
  }
  mu.eta <- function(eta)
    pmax(dPregibon(-eta, a = a, b = b),
         .Machine$double.eps^.5)
  valideta <- function(eta) TRUE
  name <- "Pregibon"
  structure(list(linkfun=linkfun, linkinv=linkinv,
    mu.eta=mu.eta, valideta=valideta, name=name),
    class = "link-glm")
}
```

Since the parameterization of the Pregibon link differs slightly from the parameterization used in the **gld** package we use the following code to define the required functions:

```
qPregibon <- function(x,a = 0,b = 0) {
  s <- (qgl(3/4,c(0,1,a-b,a+b)) -
        qgl(1/4,c(0,1,a-b,a+b)))/2.197224
  qgl(x,c(0,1,a-b,a+b))/s
}
pPregibon <- function(x,a = 0,b = 0,tol=1e-12) {
  s <- (qgl(3/4,c(0,1,a-b,a+b)) -
        qgl(1/4,c(0,1,a-b,a+b)))/2.197224
  pgl(x*s, c(0,1,a-b,a+b),inverse.eps=tol)
}
dPregibon <- function(x,a = 0,b = 0,tol=1e-12) {
  s <- (qgl(3/4,c(0,1,a-b,a+b)) -
        qgl(1/4,c(0,1,a-b,a+b)))/2.197224
  dgl(x*s, c(0,1,a-b,a+b),inverse.eps=tol)*s
}
rPregibon <- function(n,a = 0,b = 0)
    qPregibon(runif(n),a=a,b=b)
```

Note that we have fixed scale so that all members of the family have the same interquartile range; recall that scale is unidentified in this class of models.

An Example

To illustrate the fitting procedure we will consider a model of quit behavior for Western Electric workers. The transformed probability of quitting within six months of starting a new job is modeled as a linear function of a gender indicator, the score on a pre-employment dexterity exam, and a quadratic function of years of education, denoted by sex, dex, and lex, respectively. The data come originally from the study of Klein et al. (1991), but have been modified over the years to heighten the pedagogical impact.

To fit a Gosset model with a fixed value of the degrees of freedom parameter one can simply write:

```
u <- "http://www.econ.uiuc.edu/~roger/courses/
    471/data/weco.dat"
d <- read.table(u, header=TRUE)
f <- glm(kwit ~ sex + dex + poly(lex, 2),
    data=d, family=binomial(link=Gosset(1.0)))
```

This is equivalent to:

```
f <- glm(kwit ~ sex + dex + poly(lex, 2),
    data=d, family=binomial(link="cauchit"))
```

but of course the value 1.0 can be replaced with something else.

Figure 1 plots twice the log-likelihood as a function of ν for this model and indicates a 95% confidence interval for ν based on classical χ^2 limiting theory.

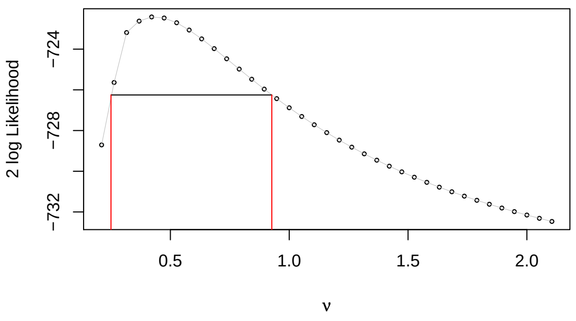


Figure 1: Profile likelihood for the Gosset link parameter ν for a model of quit behavior. The vertical lines indicate a 95% confidence interval for ν.

This interval falls strictly below one, so the Cauchit model is rejected at this conventional level of significance in favor of even heavier tailed alternative links. In light of this evidence one can, and should, still ask the question: How different are the predicted probabilities from the estimated Gosset model when compared to those from the probit specification. Figure 2 shows a PP plot comparing these fitted probabilities evaluated at the sample observations. Clearly, the two models deliver dramatically different estimates of the quit probabilities.

The Pregibon link poses somewhat more of a challenge for fitting our quit model. In Figure 3 we plot likelihood contours for the $\theta = (a, b)$ parameters of the Pregibon link. The contours are labeled according to the asymptotic relation

$$2(\ell(\hat{\theta}) - \ell(\theta)) \sim \chi^2_2,$$

where $\hat{\theta}$ is the maximum likelihood estimate. Thus, if dAIC is a difference in AIC values at the points θ and $\hat{\theta}$ then pchisq(dAIC,2) is the asymptotic p-value of a test, and confidence regions can be labeled accordingly. A formal test of the logistic hypothesis against Pregibon alternatives gives a test statistic of 14.85 with an asymptotic p-value of 0.0006, strongly repudiating the logit specification. The maximum likelihood estimates from the Pregibon model are $(\hat{a}, \hat{b}) = (-3.58, 0.57)$ suggesting a much longer tailed and somewhat skewed innovation density relative to the logistic. Figure 4 plots the fitted density.

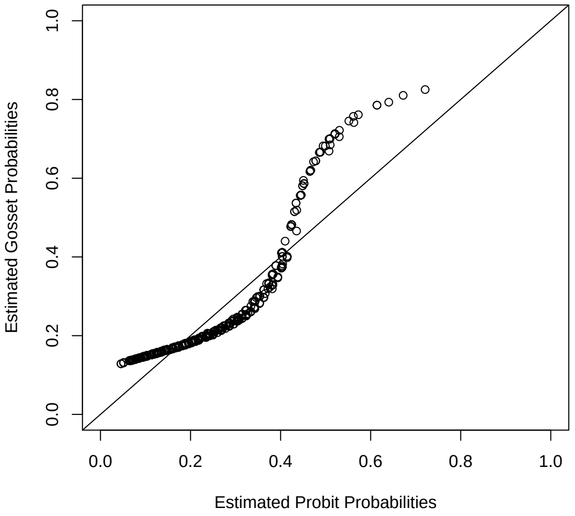


Figure 2: PP Plot of Fitted Probabilities of the Probit and Gosset Models for the WECO data: The solid line is the 45 degree line.

Evaluation of the likelihood in the region depicted in the figure requires some care to assure convergence of the glm algorithm. This is facilitated by specifying reasonable initial values of the linear predictor coefficients using the start argument. We begin the evaluation of the likelihood on the grid required for the contour plot with a central point of the grid near the maximum likelihood estimate. It is prudent to organize the remaining evaluations by moving away from the central grid point in a rectangular spiral, updating the start argument as one travels around the grid.

The code used to produce the figures is available at: http://www.econ.uiuc.edu/~roger/research/links/links.html

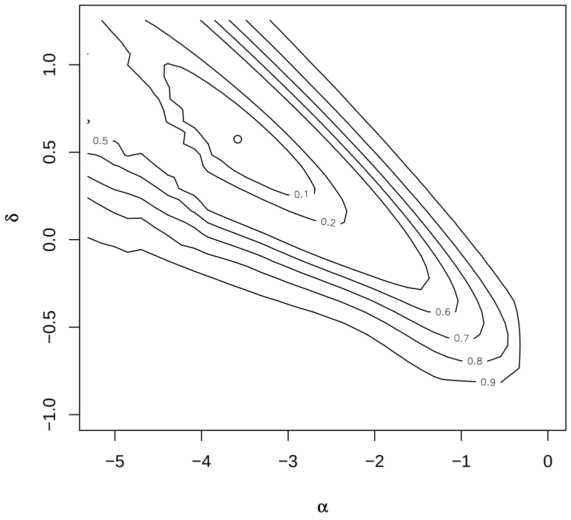


Figure 3: Profile likelihood contours for the Pregibon link parameters for a model of quit behavior. Contours are labeled according to asymptotic probability content.

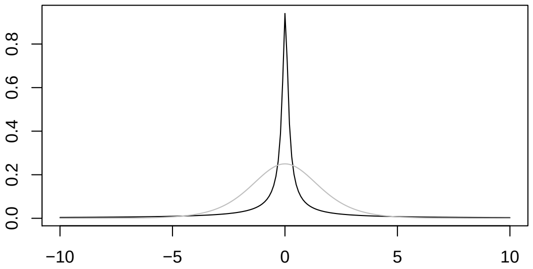


Figure 4: Fitted Pregibon innovation density for the quit model compared to the standardized logistic density (in grey).

Conclusions

Given the opportunity to estimate glm models with a wider class of parametric link functions it is obviously attractive to embed such estimation into R optimization of profiled likelihoods. This step, implemented in a somewhat perfunctory fashion in the code referred to above, finally answers my students' annual query: "What do we do if the Pregibon goodness-of-link test rejects?"

Acknowledgments

I would like to thank John Fox and Jungmo Yoon for comments that led to substantial improvements in the exposition, and I would like to thank Brian Ripley for suggestions that led to a dramatic simplification of the code.

Bibliography

R. King. *gld: Basic functions for the generalised (Tukey) lambda distribution, R package version 1.7.2*, 2005. `http://CRAN.R-project.org`.

R. Klein, R. Spady, and A. Weiss. Factors affecting the output and quit propensities of production workers. *The Review of Economic Studies*, 58:929–954, 1991.

B. J. T. Morgan and D. M. Smith. A note on Wadley's problem with overdispersion. *Applied Statistics*, 41: 349–354, 1992.

D. Pregibon. Goodness of link tests for generalized linear models. *Applied Statistics*, 29:15–24, 1980.

Roger Koenker
Department of Economics
University of Illinois
`rkoenker@uiuc.edu`

A New Package for the Birnbaum-Saunders Distribution

The bs package

by Víctor Leiva, Hugo Hernández, and Marco Riquelme

Background

Birnbaum and Saunders (1969a) derived an important lifetime distribution originating from a physical problem. The Birnbaum-Saunders distribution (BSD) describes the total time that passes until some type of cumulative damage produced by the development and growth of a dominant crack, surpasses a threshold, and causes a failure. Outside the field of the reliability, the BSD has been applied in quite a variety of fields. More details of applications and an extensive bibliographical revision can be found in Johnson et al. (1994) (on p. 651) and Vilca-Labra and Leiva (2006).

The BSD is defined in terms of the standard normal distribution through the random variable

$$T = \beta \left[\frac{\alpha Z}{2} + \sqrt{\left(\frac{\alpha Z}{2}\right)^2 + 1} \right]^2,$$

where $Z \sim \mathrm{N}(0,1)$, $\alpha > 0$, and $\beta > 0$. This is denoted by $T \sim \mathrm{BS}(\alpha, \beta)$, where α is the shape parameter and β is the scale parameter. Thus, if $T \sim \mathrm{BS}(\alpha, \beta)$,

$$Z = \frac{1}{\alpha} \left[\sqrt{\frac{T}{\beta}} - \sqrt{\frac{\beta}{T}} \right] \sim \mathrm{N}(0,1).$$

Let $T \sim \mathrm{BS}(\alpha, \beta)$. Then, the probability density function (pdf) of T is given by

$$f_T(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{1}{\alpha}\left[\sqrt{\frac{t}{\beta}}-\sqrt{\frac{\beta}{t}}\right]\right)^2} \frac{t^{-\frac{3}{2}}(t+\beta)}{2\alpha\sqrt{\beta}}, \qquad (1)$$

with $t > 0$, $\alpha > 0$, and $\beta > 0$. We note that the pdf given in (1) can be written as

$$f_T(t) = \phi(a_t(\alpha, \beta)) \frac{d}{dt} a_t(\alpha, \beta),$$

where $a_t(\alpha, \beta) = \alpha^{-1}(\sqrt{t/\beta} - \sqrt{\beta/t})$ and $\phi(\cdot)$ is the pdf of $Z \sim \mathrm{N}(0,1)$. The cumulative distribution function (cdf) of T is given by

$$F_T(t) = \Phi \left[\frac{1}{\alpha} \left(\sqrt{\frac{t}{\beta}} - \sqrt{\frac{\beta}{t}} \right) \right],$$

where $\Phi(\cdot)$ is the cdf of $Z \sim \mathrm{N}(0,1)$. Then, the quantile function (qf) of T, $t(p) = F_T^{-1}(p)$, is given by

$$t(p) = \beta \left[\frac{\alpha z_p}{2} + \sqrt{\left(\frac{\alpha z_p}{2}\right)^2 + 1} \right]^2, 0 < p < 1,$$

where z_p is the p-th percentile of $Z \sim \mathrm{N}(0,1)$. Thus, if $p = 0.5$, then $t(0.5) = \beta$, and so β is the median.

Figure 1 shows the behavior of the pdf of the BSD for some values of α. Note that as α decreases the shape of the pdf is approximately symmetrical.

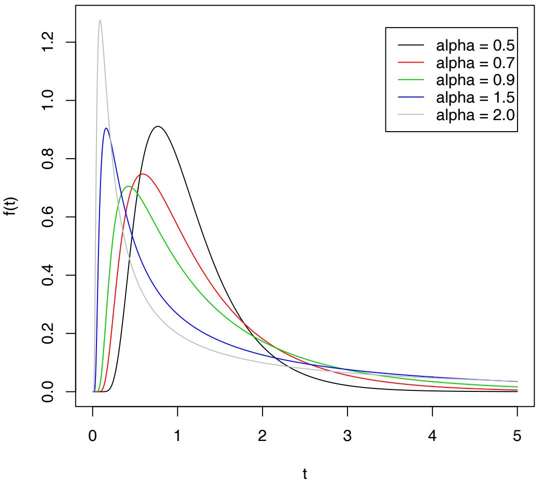


Figure 1: pdf of $T \sim \mathrm{BS}(\alpha, \beta = 1.0)$ for indicated α.

Some properties of $T \sim \mathrm{BS}(\alpha, \beta)$ are the following:
(P1) $cT \sim \mathrm{BS}(\alpha, c\beta)$, with $c > 0$.
(P2) $T^{-1} \sim \mathrm{BS}(\alpha, \beta^{-1})$.

The n-th moment of the BSD is given by $\mathbb{E}\left(T^n\right) =$

$$\beta \sum_{j=0}^{n} \binom{2n}{2j} \sum_{i=0}^{j} \binom{j}{i} \frac{(2(n-j+i))!}{2^{n-j+i}(n-j+i)!} \left(\frac{\alpha}{2}\right)^{2(n-j+i)} .$$

Thus, the mean, the variance, and the variation, skewness, and kurtosis coefficients are, respectively,

$$\begin{array}{lll}
\text{(i)} & \mu & = \frac{\beta}{2}(\alpha^2+2), \\
\text{(ii)} & \sigma^2 & = \frac{\beta^2}{4}(5\alpha^4+4\alpha^2), \\
\text{(iii)} & \gamma & = \frac{\sqrt{5\alpha^4+4\alpha^2}}{\alpha^2+2}, \\
\text{(iv)} & \delta & = \frac{44\alpha^3+24\alpha}{\left(\sqrt{5\alpha^2+4}\right)^3}, \text{and} \\
\text{(v)} & \kappa & = 3+\frac{558\alpha^4+240\alpha^2}{(5\alpha^2+4)^2}.
\end{array}$$

We note if $\alpha \to 0$, then $\delta \to 0$ and $\kappa \to 3$. The coefficients γ, δ, and κ are invariant under scale, that is, these indicators are functionally independent of the scale parameter, β.

A useful function in lifetime analysis is the failure rate (fr), defined by $h_T(t) = \frac{f_T(t)}{1-F_T(t)}$, where $f_T(\cdot)$ and $F_T(\cdot)$ are a pdf and its cdf, respectively; see, e.g., Johnson et al. (1994)(on p. 640). The BSD does not have an monotonically increasing failure rate. This is initially increasing until its critical point and then it decreases until becomes stabilized in a positive constant (not in zero), as happens with the failure rate of other distributions like the lognormal model. Specifically, if $t \to \infty$, then $h_T(t) \to (2\alpha^2\beta)^{-1}$. Other indicators of aging are the failure rate average (fra), the reliability function (rf), and the conditional reliability function (crf).

Figure 2 shows the behavior of the fr of the BSD for some values of α. Note that as α decreases the shape of the fr is approximately increasing.

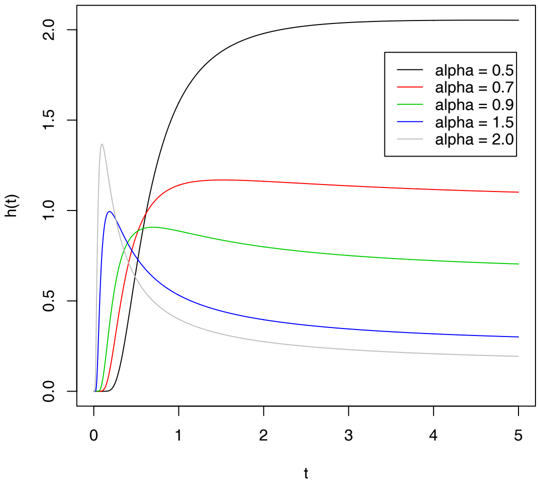


Figure 2: fr of $T \sim \mathrm{BS}(\alpha, \beta = 1.0)$ for indicated α.

This article is separated into three parts. In the second part, the main functions of the **bs** package and some illustrative examples are presented. Finally, some concluding remarks are made in the third part.

The bs functions

The **bs** package contains basic probabilistic functions, reliability indicators, and random number generators from the BSD. In order to deal with the computation of the pdf, the cdf, and the qf, the commands `dbs()`, `pbs()`, and `qbs()`, respectively, are used. The following instructions illustrate these commands.

```
> dbs(3,alpha=0.5,beta=1.0,log=FALSE)
[1] 0.02133878

> pbs(1,alpha=0.5,beta=1.0,log=FALSE)
[1] 0.5

> qbs(0.5,alpha=0.5,beta=2.5,log=FALSE)
[1] 2.5
```

For reliability analysis, in order to calculate the fr, the fra, the rf, and the crf, we have implemented the functions `frbs()`, `frabs()`, `rfbs()`, and `crfbs()`, respectively. For obtaining random numbers, we have developed the following three generators based on: (G1) the relationship between the standard normal distribution (SND) and the BSD (see (Chang and Tang, 1994a)); (G2) the relationship between the sinh-normal distribution (SHND) and the BSD (see (Rieck, 2003)); (G3) the relationship between the inverse Gaussian distribution (IGD) and the BSD (Johnson et al. (1994), p. 658). For these three generators, the functions rbs1(), rbs2(), and rbs3() are used, respectively. Also, we have developed a common function, named rbs(), for obtaining random numbers from the suitable generator depending on the desired setting. The function rbs() selects the most appropriate method automatically. For details about effectiveness and efficiency of these three generators, and in order to select the most suitable one see Leiva, Sanhueza et al. (2006). The following instructions illustrate these commands:

```
> rbs3(n=6,alpha=0.5,beta=1.0)
[1] 0.7372910 0.4480005 1.8632176
[4] 0.9728011 1.2675537 0.2252379

> rbs(n=6,alpha=0.5,beta=1.0)
[1] 0.5905414 1.1378133 1.1664306
[4] 1.3187935 1.2609212 1.8212990
```

Another group of functions related to estimation, graphical analysis, and goodness-of-fit for the BSD are also available.

In order to estimate the shape (α) and scale (β) parameters from the BSD, we have implemented the following three methods based on: (E1) the likelihood method (MLE) and the mean-mean estimator (see (Birnbaum and Saunders, 1969b)); (E2) a graphical method (GME), which permits estimation of α and β by the least square method (see (Chang and Tang, 1994b)); (E3) the modified moment method (MME) (see (Ng et al., 2003)). For E1, E2, and E3, the functions est1bs(), est2bs(), and est3bs() are used. Next, two examples related to the use of these commands are presented. The first example is based on simulated data and the second example is from a real data set.

Example 1. In order to carry out simulation studies, we develop the functions simul.bs.gme(), simul.bs.mle(), and simul.bs.mme(). These functions generate random samples, estimate parameters, and establish goodness-of-fit. The samples of size n, one for each method (G1, G2, or G3), are generated by using rbs1(), rbs2(), and rbs3(), respectively. The estimations, one for each method (E1, E2, or E3), are obtained by using est1bs(), est2bs(), and est3bs(), respectively. The goodness-of-fit method is based on the statistic of Kolmogorov-Smirnov (KS), which is available by means of the function ksbs(). The generated observations by means of G1, G2, and G3 are saved as slots of the R class simulBsClass, which are named sample1, sample2, and sample3, respectively. Furthermore, the results of the simulation study are saved in a fourth slot of this class, named results. Thus, for instance, the instruction

```
> simul.bs.mle(100,0.5,1.0)
```

simulates three samples of size $n = 100$ from a population $T \sim \text{BS}(\alpha = 0.5, \beta = 1.0)$, one for each method (G1, G2, or G3), computes the MLES's for α and β, and establish goodness-of-fit for each sample. The results can be saved in the variable resultsOfsimul.bs.mle by using the instruction

```
> resultsOfsimul.bs.mle<-
                    simul.bs.mle(100,0.5,1.0)
```

Hence,

```
> resultsOfsimul.bs.mle@results
```

retrieves the slot results, which gives the following summary of the simulation study:

```
        MLE(alpha) MLE(beta)  KS      p-value
Sample1  0.66456    1.06143   0.06701 0.76032
Sample2  0.43454    0.93341   0.09310 0.35135
Sample3  0.60549    0.86124   0.08475 0.46917
```

In addition,

```
> resultsOfsimul.bs.mle@sample1
```

retrieves the slot sample1, which only shows the generated sample associated with G1 and does not show the estimates neither the goodness-of-fit.

Next, we give an example of real data in order to illustrate the functions est1bs(), est2bs(), and est3bs().

Example 2. Birnbaum and Saunders (1969b) reported a data set (see Table 1) corresponding to fatigue life (T) measures in cycles ($\times 10^{-3}$) of $n = 101$ aluminum coupons (specimens) of type 6061-T6. These specimens were cut in a parallel angle to the rotation way and oscillating to 18 cycles per seconds. The coupons were exposed to a pressure with maximum stress of 31.000 psi (pounds per square inch). All specimens were tested until failure.

The **bs** package brings these loaded data, so that if we input the command psi31, the data will be ready to use them. Thus,

```
> data <- psi31
```

saves the data set in the variable data. The instruction

```
> estimates <- est1bs(data)
```

computes the MLE's for α and β and uses the mean-mean-estimate (see (Birnbaum and Saunders, 1969b)) as initial estimator for β. Then, the following result is obtained:

```
> estimates
$beta.starting.value [1] 131.8193

$alpha [1] 0.1703847

$beta [1] 131.8188

$converge [1] "TRUE"

$iteration [1] 2
```

Table 1: Lifetimes of aluminum specimens exposed to a maximum stress of 31.000 psi.

70	90	96	97	99	100	103	104
104	105	107	108	108	108	109	109
112	112	113	114	114	114	116	119
120	120	120	121	121	123	124	124
124	124	124	128	128	129	129	130
130	130	131	131	131	131	131	132
132	132	133	134	134	134	134	134
136	136	137	138	138	138	139	139
141	141	142	142	142	142	142	142
144	144	145	146	148	148	149	151
151	152	155	156	157	157	157	157
158	159	162	163	163	164	166	166
168	170	174	196	212			

The estimations of α and β can be saved in the variables alpha and beta, that is,

```
> alpha <- estimate$alpha
> beta <- estimate$beta
```

Also, by using the invariance property of the MLE's, we obtain estimations for the mean, the variance, and the variation, skewness, and kurtosis coefficients. The function indicatorsbs() computes the MLE's for α, β, μ, σ^2, γ, δ, and κ. Thus, the instruction

```
> indicatorsbs(data)
```

gives the following results:

```
The MLE's are:
Alpha = 0.1703847
Beta  = 131.8188
Mean = 143.0487
Variance = 522.753
Variation coefficient = 0.170967
Skewness coefficient = 0.5103301
Kurtosis coefficient = 3.43287
```

Another five data sets, called psi21, psi26, bearings, timerepair, and biaxial, used frequently in the literature of this topic have also been incorporated in the **bs** package.

Next, the implemented graphical functions will be described. Firstly, the command graphdbs() allows visualization of five different pdf's from the BSD, but the legend of this must be added. Figure 1 was plotted with this command by means of the instructions

```
> graphdbs(0.5,0.7,0.9,1.5,2.0,1.0,1.0,1.0,
  1.0,1.0)
```

and

```
> legend(3.45,2,c("alpha = 0.5","alpha = 0.7",
  "alpha = 0.9","alpha = 1.5","alpha = 2.0"),
  lty=c(1,1,1,1,1), col=c(1,2,3,4,8))
```

Figure 2 was plotted with the function grapfrbs(), which is similar to graphdbs(), but uses the visualization of five different fr from the BSD. Secondly, the command qqbs() allows drawing a quantile-quantile (Q-Q) plot to check graphically if a set of observations follows a particular BSD. This command also incorporates a straight line Q-Q that sketches a line passing through the first and the third quartile of the theoretical BSD. The instruction

```
> qqbs(data,alpha,beta,line=TRUE)
```

gives a Q-Q plot for psi31. Figure 3 shows this graph.

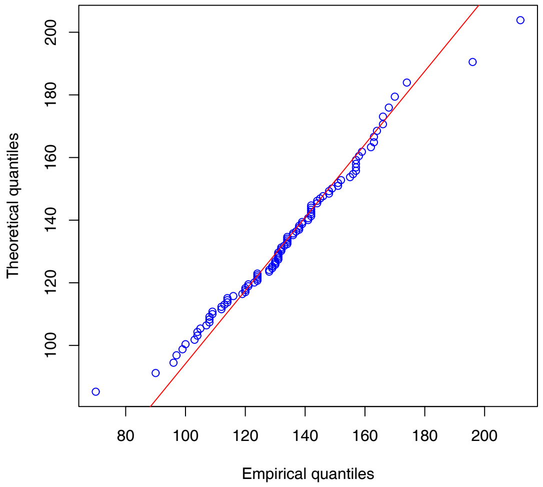


Figure 3: Q-Q plot for the BSD for psi31.

Thirdly, the command gmbs() allows simultaneously estimating α and β by GME and drawing the Q-Q type plot by Chang and Tang (1994b). The estimations are summarized in Table 2.

Figure 4 shows a graph produced by using the instruction gmbs(data).

Table 2: Estimates of α and β by using `gmbs(data)`, for the data in Table 1.

$\widehat{\alpha}$	$\widehat{\beta}$	R^2
0.1686	131.9224	0.9775

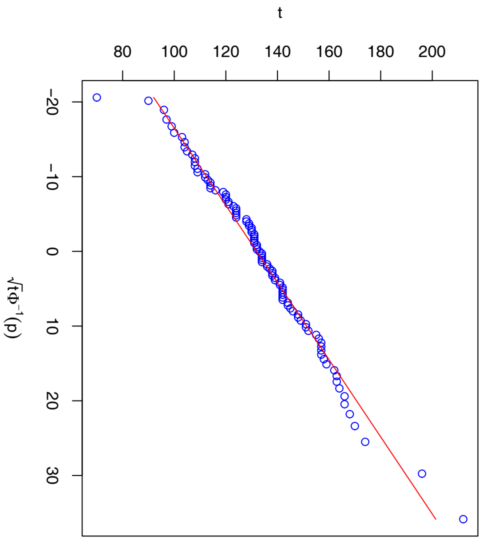


Figure 4: fitted plot for the BSD by using graphical method for psi31.

We also have implemented commands for computing methods of goodness-of-fit and criteria of model selection from the BSD. These functions are `ksbs()`, `sicbs()`, `aicbs()`, and `hqbs()`, which calculate the KS test, the Schwartz information criterium (SIC), the Akaike information criterium (AIC), and the Hannan-Queen information criterium (HQC). The command `ksbs()` gives also a comparative graph of the cdf and the empirical distribution function. Next, the results for `psi31` are presented in Table 3.

Finally, by using methods of goodness-of-fit based on moments, we have implemented the β_1-β_2 and δ_2-δ_3 charts; see Barros (2006). These charts-of-fit are particularly useful when various data sets are collected. For example, in environmental sciences we frequently found random variables measured hourly at different sampling points. Then, for every year we have a monthly data set available at each sampling point. Thus, it is possible to compute estimations for β_1 (the square of skewness coefficient), β_2 (the kurtosis coefficient), δ_2 (the square of variation coefficient), and δ_3 (the skewness coefficient), for every month at each sampling point. In this way, the pairs $(\widehat{\beta}_1, \widehat{\beta}_2)$ and $(\widehat{\delta}_2, \widehat{\delta}_3)$ are plotted inside the theoretical β_1-β_2 and δ_2-δ_3 charts, respectively. The functions `fitbetabs()` and `fitdeltabs()` have been developed to add these charts to the scatter plot of the pairs $(\widehat{\beta}_1, \widehat{\beta}_2)$ and $(\widehat{\delta}_2, \widehat{\delta}_3)$, respectively. In order to illustrate this methodology, we have simulated twelve (12) samples by using the command `rbs(n=30,alpha,beta=1.0)`, with $\alpha = 0.2(0.2)2.4$, which can represent, for example, daily data for every month during one calendar year. These result have been saved in the matrix `samplei`, with $i = 1, \ldots, 12$. Next, the estimations of δ_2, δ_3, β_1, and β_2 are obtained and saved in the vectors `x` and `y`. Finally, the data sets have been fitted to the charts. Thus, by using the instructions

```
> plot(x,y,xlab=expression(beta*1),
  ylab=expression(beta*2),col=4)
> fitbetabs(0.2,2)
> fitbetabs(2.4,3)
```

and

```
> plot(x,y,xlab=expression(delta*2),
  ylab=expression(delta*3),col=4)
> fitdeltabs(0.2,2)
> fitdeltabs(2.4,3)
```

we have obtained the following graphs:
BS$(\alpha = 0.2, \beta = 1.0)$ in red;
BS$(\alpha = 2.4, \beta = 1.0)$ in green;
and pairs (x, y) in blue]:

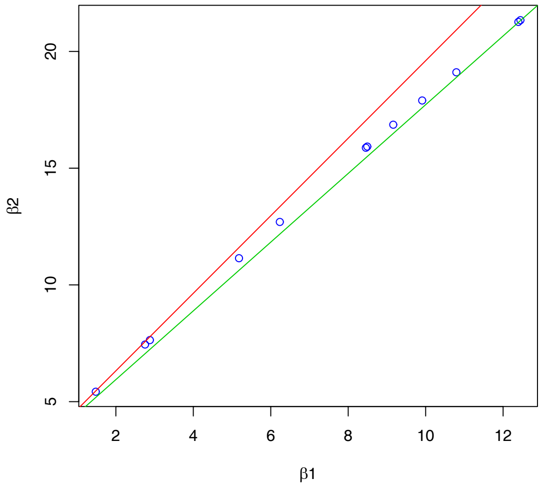


Figure 5: chart β_1-β_2 for twelve simulated samples.

Table 3: Table 3: SIC, AIC, HQC, and KS test for psi31.

SIC	AIC	HQC	KS (p-value)
4.573125	4.547233	4.557715	0.085 ($p = 0.459$)

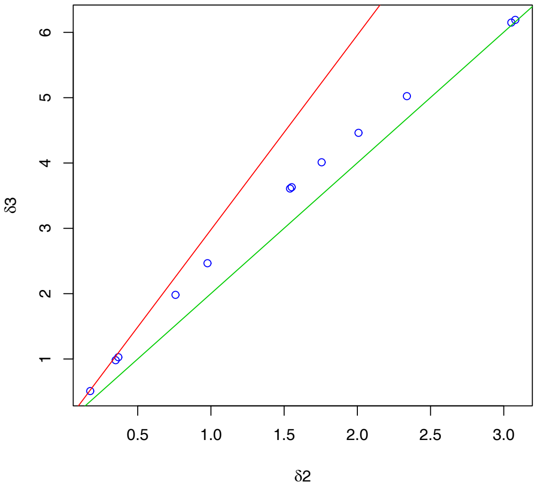


Figure 6: chart δ_2-δ_3 for twelve simulated samples.

Concluding remarks

In order to analyze data coming from the BSD, we have developed the new **bs** package, which implements several useful commands. The created functions are related to probability and reliability indicators, estimation, goodness-of-fit, and simulation. In addition, some commands related to graphical tools were also developed. We think that in the future, the **bs** package can be improved by incorporating other functions related to the estimation for censored data, regression and diagnostic methods, as well as generalizations of the BSD. The theoretical aspects of this last part have been published by Galea et al. (2004), Díaz-García and Leiva (2005), Vilca-Labra and Leiva (2006), Leiva, Barros et al. (2006).

Acknowledgments

This study was carried out with the support of research projects FONDECYT 1050862, FANDES C-13955(10), DIPUV 42-2004, and DIPUCM 200601, Chile. This article was partially written during the time that Dr. Víctor Leiva was a Visiting Scholar in the University of North Carolina (UNC) at Chapel Hill, USA. The authors are thankful to the Associate Editor and to the referee for his truly helpful suggestions which led to a much improved presentation of this article.

Bibliography

M. Barros, V. Leiva, and G.A. Paula. Moments of the generalized Birnbaum–Saunders distribution: specification and application. *Submitted to Publication*, 2006.

Z.W. Birnbaum and S.C. Saunders. A new family of life distributions. *Journal of Applied Probability*, 6(2): 637-652, 1969a.

Z.W. Birnbaum and S.C. Saunders. Estimation for a family of life distributions with applications to fatigue. *Journal of Applied Probability*, 6(2):328-347, 1969b.

D.S. Chang and L.C. Tang. Random number generator for the Birnbaum-Saunders distribution. *Computational and Industrial Engineering*, 27(1-4):345-348, 1994a.

D.S. Chang and L.C. Tang. Graphical analysis for Birnbaum-Saunders distribution. *Microelectronics and Reliability*, 34(1):17-22, 1994b.

J.A. Díaz-García and V. Leiva. A new family of life distributions based on the elliptically contoured distributions. *Journal of Statistical Planning and Inference*, 128(2):445-457, 2005.

M. Galea, V. Leiva, and G.A. Paula. Influence diagnostics in log-Birnbaum-Saunders regression models. *Journal of Applied Statistics*, 31:9, 1049-1064, 2004.

N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions: Volume 2*. Wiley, New York, 1994.

V. Leiva, M. Barros, G.A. Paula, and M. Galea. Influence diagnostics in log-Birnbaum-Saunders regression models with censored data. Accepted in *Computational Statistics and Data Analysis*, 2006.

V. Leiva, A. Sanhueza, P.K. Sen, and G.A. Paula. Random number generators for the generalized Birnbaum-Saunders distribution. *Submitted to Publication*, 2006.

H.K.T. Ng, D. Kundub, and N. Balakrishnan. Modified moment estimation for the two-parameter Birnbaum-Saunders distribution. *Computational Statistics and Data Analysis*, 43(2):283-298, 2003.

J.R. Rieck. A comparison of two random number generators for the Birnbaum-Saunders distribution. *Communications in Statistics - Theory and Methods*, 32(5):929-934, 2003.

F. Vilca-Labra and V. Leiva. A new fatigue life model based on the family of skew-elliptical distributions. *Communications in Statistics: Theory and Methods*, 35(2):1-16, 2006.

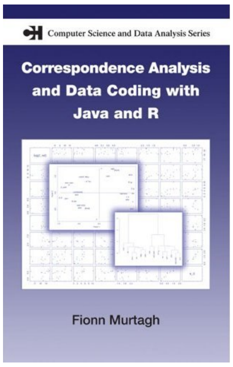
Víctor Leiva
Departamento de Estadística, CIMFAV
Universidad de Valparaíso, Chile
victor.leiva@uv.cl; victor.leiva@yahoo.com

Hugo Hernández
Departamento de Estadística
Universidad de Valparaíso, Chile
hhp_uv@yahoo.com

Marco Riquelme
Instituto de Ciencias Básicas
Universidad Católica del Maule, Chile
mriquelm@ucm.cl

Review of Fionn Murtagh's book: Correspondence Analysis and Data Coding with Java and R

by Susan Holmes


Computer Science and Data Analysis Series
Correspondence Analysis and Data Coding with Java and R
Fionn Murtagh

Correspondence analysis is a method for analysing contingency tables, in particular two-way tables, by decomposing the Chi-square distances between the rows in much the same way that principal component analysis (PCA) decomposes the variance of a multivariate continuous matrix. For the reader familiar with multidimensional scaling (Mardia et al., 1979; Ripley, 1996), this can also be seen as a way of representing Chi-square distances between the rows or the columns of the contingency table. This method has been rediscovered regularly and interested readers can read the history of the method in Benzécri (1982) and de Leeuw (1983).

For those familiar with the matrix formulation of PCA as the singular value decomposition of $X_{n\times p}$ as the product of orthogonal matrices U and V with a diagonal matrix of decreasing singular values S, $X = USV'$, this provides the best rank k approximation to X by just taking the first k columns of U. To get correspondence analysis, replace X by the contingency table divided by its total (call this F, the frequency table), call the row sums of F $\mathbf{r}$ and the column sums $\mathbf{c}$. Let the diagonal matrices defined by these vectors be $\mathbf{D}_r$ and $\mathbf{D}_c$. Write the singular value decomposition of

$$\mathbf{D}_r{}^{-\frac{1}{2}}F\mathbf{D}_c{}^{-\frac{1}{2}} = USV'$$

with $V'V = I, U'U = I$; then $\mathbf{D}_r{}^{-1}F\mathbf{D}_c{}^{-1}$ can be written ASB', with $A = \mathbf{D}_r{}^{-\frac{1}{2}}U$ and $B = \mathbf{D}_c{}^{-\frac{1}{2}}V$. One can verify that $A'D_rA = I$ and $B'D_cB = I$; see Holmes (2006) for more details. This decomposition can be used to provide a graphical representation of both columns and rows that is meaningful even when both are plotted on the same graph. This biplot representation is a useful feature of the method.

What the book contains

The book starts by setting the stage for the book's particular perspective with a foreword by Jean-Paul Benzécri, a leader in the world of geometric approaches to multivariate analyses. Chapter 2 covers the advent of France's particular school of what was called 'Analyse des Données' to distinguish it from classical parametric statistics based on probability theory. The insistence of Murtagh's mentor, Jean-Paul Benzécri on the data over the model is a precursor to modern day machine learning and a European parallel to John Tukey's Exploratory Data Analysis.

After the introduction, chapter 3 gives a complete matrix-based explanation of correspondence analysis followed by chapter 4 on the importance of choosing the correct encoding of data, and chapter 5 presents examples. Chapter 6 is dedicated to the specific study of textual analyses.

Fionn Murtagh aims to explain how to effectively analyse multivariate categorical data. He presents both the different ways of encoding the data and ways of analysing them following the French approach to correspondence analysis. Many parts of the book are also of historical interest: the introduction by Benzecri and the fact that the programs were coded from his original Pascal programs for instance. His text contains many examples especially from the field of textual analysis, closely following his mentor.

The Rebirth of Correspondence Analysis

The book is somewhat old fashioned and doesn't present bridges to today's literature. Correspondence analysis is regularly rediscovered, and today's literature on the decomposition of the structure of the web is no exception. An interesting connection between Kleinberg's hubs and authorities (Kleinberg, 1999) and correspondence analysis is pointed out by Fouss et al. (2004). This should motivate modern readers to understand something that lies at the heart of successes such as Google's page-rank system. In fact, a good example of what profiles are can be provided by looking at Google's Trends feature, for which the towns are given as profiles divided by the number of searches for that city. Try googling yourself [1].

Missing Topics

It's a shame that the author didn't take the opportunity to explain in more detail to the English speaking audience the French community's special tricks for interpreting correspondence analysis. Although there are special programs for supplementary variables and individuals in the book (from page 59 on), and they are mentioned on pages 41 and 43, no example shows clearly how they are used; this would have been an important contribution to the existing literature.

Important Caveats

The uninitiated user should be cautioned as to how important it is to compare eigenvalues at the outset of the analysis. Scree plots of the eigenvalues have to be made before choosing the number of axes, and the big mistakes that occur in CA happen when two close eigenvalues are split. There may be instances when not separating three axes with three similar eigenvalues would encourage use of tools such as `xgobi` and `ggobi` for dynamic multidimensional graphics.

The book would also be improved by more background about special aspects of correspondence analysis, such as Guttman's horseshoe effect and seriation (Charnomordic and Holmes, 2001), for which correspondence analysis is especially useful.

The present reviewer's frustration with this book comes mostly from the fact that there are many good 'First Books on Correspondence Analaysis' available in English (Greenacre, 1984; Lebart et al., 1984). There are some first-rate R packages for performing correspondence analysis that we list at the end of this review. However there is no good followup text that carefully explains useful techniques developed in the literature such as internal correspondence analysis (Cazes et al., 1988), asymmetric correspondence analysis, conjoint analysis of several contingency tables through ACT (Lavit et al., 1994) or through cocorrespondence analysis (ter Braak, 1986).

Software implementation

As regards the software implementation provided, it does not seem a good use of paper to present the source code in the main text, but providing a complete R package on CRAN would definitely be a worthwhile investment. The R code as it exists is not modernised and doesn't use classes or object oritentation.

The Java code was not useful to the present reviewer as on our modern machine (Mac running Mac OS X 10.4) the instructions provided on the website do not allow one to execute the JAVA downloaded.

Packages containing other R correspondence analysis programs available at CRAN

`ade4` : The most complete suite of multivariate functions 'à la française', this package includes internal correspondence analysis, asymmetric correspondence analysis, detrended correspondence analysis and many visualization tools. Its philosophy and class structures are based on the triplets defined as duality diagrams in Escoufier (1977) and discussed more recently (Holmes, 2006). A description of the simpler methods (one table methods) is available in English in Chessel et al. (2004).

`amap` The function `afc` performs the classical correspondence analysis.

`CoCorresp` Package for doing correspondence analysis on several tables and co-inertia analyses especially used by ecologists.

`ca` **and** `homals` from Jan de Leeuw's Psychometrics with R bundle (`http://www.cuddyvalley.org/psychoR/`).

`vegan` Developed by plant ecologists, this package allows one to perform constrained as well as vanilla correspondence analysis in the function `cca`; it also provides much needed detrending for gradients in the `decorana` function, which

[1] `http://www.google.com/trends?q=statistical+software\&ctab=0\&geo=all\&date=2005`

implements Hill and Gauch's detrending technique (Hill and Gauch, 1980).

`VR` Venables and Ripley bundle containing MASS. Within MASS is the bare-bones function `corresp` which allows one to produce a correspondence analysis map and use the `biplot` function to plot it.

Bibliography

J.-P. Benzécri. *Histoire et préhistoire de l'analyse des données*. Dunod, 1982.

P. Cazes, D. Chessel, and S. Dolédec. L'analyse des correspondances internes d'un tableau partitionné: son usage en hydrobiologie. *Revue de Statistique Appliquée*, 36:39–54, 1988.

B. Charnomordic and S. Holmes. Correspondence analysis for microarrays. *Statistical Graphics and Computing Newlsetter*, 12, 2001.

D. Chessel, A. B. Dufour, and J. Thioulouse. The ade4 package — I: One-table methods. *R News*, 4(1):5–10, 2004. URL http://CRAN.R-project.org/doc/Rnews/.

J. de Leeuw. On the prehistory of correspondence analysis. *Statistica Neerelandica*, 37:161–164, 1983.

Y. Escoufier. Operators related to a data matrix. In J. R. Barra, editor, *Recent Developments in Statistics.*, pages 125–131. North Holland, 1977.

F. Fouss, J.-M. Renders, and M. Saerens. Some relationships between Kleinberg's hubs and authorities, correspondence analysis, and the Salsa algorithm. In *JADT 2004, International Conference on the Statistical Analysis of Textual Data*, pages 445–455, Louvain-la-Neuve, 2004.

M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, London., 1984.

M. Hill and H. Gauch. Detrended correspondence analysis, an improved ordination technique. *Vegetatio*, 42:47–58, 1980.

S. Holmes. Multivariate analysis: The French way. In *Festschrift for David Freedman*, Beachwood, OH, 2006. (edited by D. Nolan and T. Speed) IMS.

J. M. Kleinberg. Hubs, authorities, and communities. *ACM Computing Surveys*, 31(4es):5, 1999. ISSN 0360-0300.

C. Lavit, Y. Escoufier, R. Sabatier, and Traissac. The ACT (STATIS method). *Computational Statistics & Data Analysis*, 18:97–119, 1994.

L. Lebart, A. Morineau, and K. M. Warwick. *Multivariate Descriptive Statistical Analysis*. Wiley, 1984.

K. Mardia, J. Kent, and J. Bibby. *Multiariate Analysis*. Academic Press, NY., 1979.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

C. J. F. ter Braak. Canonical correspondence analysis: a new eigenvector method for multivariate direct gradient analysis. *Ecology*, 67:1167–1179, 1986.

Susan Holmes
Statistics Department
Stanford, CA 94305
susan@stat.stanford.edu

R Help Desk

Accessing the Sources

by Uwe Ligges

Introduction

One of the major advantages of open source software such as R is implied by its name: the sources are open (accessible) for everybody.

There are many reasons to look at the sources. One example is that a user might want to know exactly what the software does, but the documentation is not sufficiently explicit about the underlying algorithm. As another example, a user might want to change some code in order to implement a modification of a given (already implemented) method or in order to fix a bug in a contributed package or even in R itself.

How to access different kinds of sources (in order to read or to change them), both in R itself and in packages, is described in the following sections.

It is always a good idea to look into appropriate manuals for your current R version, if working on the sources is required. Almost all manuals contain relevant information for this topic: 'An Introduction to R', 'Writing R Extensions', 'R Installation and Administration', and 'The R Language Definition' (Venables et al., 2006; R Development Core Team, 2006a,b,c).

R Code Sources

In most cases, it is sufficient to read some R code of the function in question and look at how other functions are called or how the data are manipulated within a function. The fastest and easiest way to do so for simple functions is to type the function's name into R and let R print the object. For example, here is how to view the source code for the function `matrix()`:

```
> matrix
function (data = NA, nrow = 1, ncol = 1,
          byrow = FALSE, dimnames = NULL)
{
    data <- as.vector(data)
    if (missing(nrow))
        nrow <- ceiling(length(data)/ncol)
    else if (missing(ncol))
        ncol <- ceiling(length(data)/nrow)
    x <- .Internal(matrix(data, nrow, ncol,
                          byrow))
    dimnames(x) <- dimnames
    x
}
<environment: namespace:base>
```

Unfortunately, comments in the code may have been removed from the printed output, because they were already removed in the loaded or installed package in order to save memory. This is *in principle* controlled by the arguments `keep.source` (for R) and `keep.source.pkgs` (for packages) of the `options()` function. If these arguments are set to `TRUE` comments are not removed, but there are also a lot of circumstances, e.g., lazy-loading and saved images of packages (Ripley, 2004), where setting these arguments to `TRUE` does not help to keep comments in the sources.

Therefore, it makes more sense to look into the original sources, i.e., those from which the package or R have been installed. Both R sources and sources of contributed packages are available from the CRAN mirrors (see `http://cran.r-project.org/mirrors.html` for a current list of mirrors).

After unpacking a source package, the R source code can be found in the directory `PackageName/R/`. Since the files therein are plain text files, they can be opened in the user's favorite text editor. For R's base packages, the R code is available in R's subdirectory `$R_HOME/src/library/PakageName/R/`. For package bundles, replace `PackageName` by `BundleName/PackageName` in the paths given above.

Code Hidden in a Namespace

In some cases, a seemingly missing function is called within another function. Such a function might simply be hidden in a namespace (Tierney, 2003). Type `getAnywhere("FunctionName")` in order to find it. This function reports which namespace a function comes from, and one can then look into the sources of the corresponding package. This is particularly true for S3 methods such as, for example, `plot.factor`:

```
R> plot.factor
Error: object "plot.factor" not found

R> getAnywhere("plot.factor")
A single object matching 'plot.factor' was found
It was found in the following places
  registered S3 method for plot from namespace
    graphics
  namespace:graphics
with value
### [function code omitted] ###
```

The file that contains the code of `plot.factor` is '$R_HOME/src/library/graphics/R/plot.R'.

S3 and S4

As another example, suppose that we have the object `lmObj`, which results from a call to `lm()`, and we would like to find out what happens when the object is printed. In that case, a new user probably types

```
R> print
```

in order to see the code for printing. The frustrating result of the call is simply:

```
function (x, ...)
UseMethod("print")
<environment: namespace:base>
```

The more experienced user knows a call to `UseMethod()` indicates that `print()` is an S3 generic and calls the specific method function that is appropriate for the object of class `class(x)`. It is possible to ask for available methods with `methods(print)`. The function of interest is the S3 method `print.lm()` from namespace **stats** (the generic itself is in the **base** package namespace). A method hidden in a namespace can be accessed (and therefore printed) directly using the `:::` operator as in `stats:::print.lm`. The file containing the source code for printing the `lmObj` object is available from package **stats**' sources: '$R_HOME/src/library/stats/R/lm.R'.

In order to understand and change S4 related sources, it is highly advisable to work directly with a package's source files. For a quick look, functions such as `getClass()`, `getGeneric()`, and `getMethod()` are available. The following example prints the code of the `show()` method for `mle` objects from the **stats4** package:

```
R> library("stats4")
R> getMethod("show", "mle")
```

Compiled Code Sources

When looking at R source code, sometimes calls to one of the following functions show up: `.C()`, `.Call()`, `.Fortran()`, `.External()`, or `.Internal()`

and .Primitive(). These functions are calling entry points in compiled code such as shared objects, static libraries or dynamic link libraries. Therefore, it is necessary to look into the sources of the compiled code, if complete understanding of the code is required.

In order to access the sources of compiled code (i.e., C, C++, or Fortran), it is not sufficient to have the binary version of R or a contributed package installed. Rather, it is necessary to download the sources for R or for the package. How to download those sources is described in the previous section.

For contributed packages, source code can be found in the directory PackageName/src/ (again, for package bundles replace PackageName by BundleName/PackageName). Files therein can be searched for the corresponding entry point.

For R and standard R packages, compiled code sources are in subdirectories of $R_HOME/src/. Particularly $R_HOME/src/main/ contains most of the interesting code. The first step is to look up the entry point in file '$R_HOME/src/main/names.c', if the calling R function is either .Primitive() or .Internal(). This is done in the following example for the code implementing the 'simple' R function sum().

Typing sum shows the R code of that function:

```
R> sum
function (..., na.rm = FALSE)
.Internal(sum(..., na.rm = na.rm))
<environment: namespace:base>
```

Obviously, there is only *one* relevant call within sum(), namely the .Internal call to an inner entry point sum(). The next step is to look up that entry point in the file 'names.c', which reveals the following line:

```
/* many lines omitted */
  {"sum", do_summary, 0, 11, -1,
                   {PP_FUNCALL, PREC_FN, 0}},
/* many lines omitted */
```

This line tells us to look for do_summary which itself lives in file 'summary.c' in the same directory. If the filename is not obvious, then it can be found by simply 'grep'ping for the string in R's $R_HOME/src/ path.

The Bleeding Edge

Folks working on the bleeding edge of statistical computing might want to check out the most recent sources, e.g., by looking into the current svn archives of R. To access them via a web browser, visit https://svn.r-project.org/R/. The subdirectory ./trunk/ contains the current R-devel version; other branches (such as R-patched) can be found in ./branches/, and released versions of R can be found in ./tags/.

Summary

It is easy to look at both R and C, C++ or Fortran sources. It is not that difficult to change the sources and to recompile or to reinstall a modified package. This way, users can become developers very easily and contribute bugfixes and new features to existing packages or even to R itself.

Acknowledgments

I would like to thank Roger Bivand, Gabor Grothendieck, Paul Murrell, and David Scott for suggestions to improve this Help Desk article.

Bibliography

R Development Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2006a. URL http://www.R-project.org.

R Development Core Team. *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, Austria, 2006b. URL http://www.R-project.org.

R Development Core Team. *R Language Definition*. R Foundation for Statistical Computing, Vienna, Austria, 2006c. URL http://www.R-project.org.

B. Ripley. Lazy Loading and Packages in R 2.0.0. *R News*, 4(2):2–4, September 2004. ISSN 1609-3631. URL http://CRAN.R-project.org/doc/Rnews/.

L. Tierney. Name Space Management for R. *R News*, 3(1):2–6, June 2003. ISSN 1609-3631. URL http://CRAN.R-project.org/doc/Rnews/.

W. N. Venables, D. M. Smith, and the R Development Core Team. *An Introduction to R*. R Foundation for Statistical Computing, Vienna, Austria, 2006. URL http://www.R-project.org.

Uwe Ligges
Fachbereich Statistik, Universität Dortmund, Germany
ligges@statistik.uni-dortmund.de

useR! 2006, The Second R User Conference

by Balasubramanian Narasimhan

The second useR! 2006 conference (http://www.R-project.org/useR-2006) took place in Vienna, June 15–17. The conference was organized by the Austrian Association for Statistical Computing and Wirtschaftsuniversität Wien and sponsored by the R foundation for Statistical Computing as well as the American Statistical Association section on Statistical Computing. The organizing committee consisted of Torsten Hothorn, Achim Zeileis, David Meyer, and Bettina Grün (in charge of conference, program, local organization, and web site, consecutively) along with Kurt Hornik and Friedrich Leisch.

The approximately 400 attendees comprised a broad spectrum of users from the scientific and business community. The conference was divided into keynote, kaleidoscope, and focus sessions. The kaleidoscope sessions addressed applications and usage of R appealing to a broad audience, while the focus sessions were more specialized, highlighting a specific topic followed by a discussion and exhibition. Keynote speakers included R core members and R users:

- John Chambers on the history of S and R
- Brian Everitt on cluster analysis
- John Fox and Sanford Weisberg on using R for teaching
- Trevor Hastie on path algorithms
- Jan de Leeuw on R in psychometrics
- Uwe Ligges, Stefano Iacus and Simon Urbanek on R on various platforms
- Paul Murrell on drawing graphs in R
- Peter Rossi on Bayesian statisics with marketing data in R

The conference web page hosts abstracts and slides for all presentations, highlighting the role R is playing in many fields from teaching to research to commercial applications. The conference concluded with a panel discussion on getting recognition for excellence in computational statisics. The panelists included Jan de Leeuw (Editor, Journal of Statisical Software), Timothy Hesterberg (Past-Chair, ASA Section on Statistical Computing), Martina Mittlböck, (Associate Editor, Computational Statistics & Data Analysis), Paul Murrell (Chair, ASA Section on Statistical Graphics), Erich Neuwirth (Associate Editor, Computational Statistics), and Luke Tierney (Editor, Journal of Computational and Graphical Statistics). A video of the panel discussion is available from the website and is recommended viewing.

I would be remiss if I did not mention the routinely excellent arrangements made by the organizers. The conference provided a wonderful milieu for discussions over coffee breaks, lunches, pub outings and dinner, and judging by the activity, it was a roaring success. Social events included a cocktail reception at the Vienna City Hall (Rathaus), and a traditional Viennese dinner at Heuriger Fuhrgassl-Huber restaurant with generous sampling of Viennese white wine. Everyone was enthusiastic in praise of the conference and I am sure that the next useR! is eagerly anticipated.

Balasubramanian Narasimhan
Stanford University, USA
naras@stanford.edu

Changes in R

by the R Core Team

User-visible changes

- The startup message now prints first the version string and then the copyright notice (to be more similar to `R --version`).
- `save()` by default evaluates promise objects. The old behaviour (to save the promise and its evaluation environment) can be obtained by setting the new argument `'eval.promises'` to FALSE. (Note that this does not apply to promises embedded in objects, only to top-level objects.)
- The functions `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()` now default their `'comment.char'` argument to `""`. (These functions are designed to read files produced by other software, which might use the # character inside fields, but are unlikely to use it for comments.)
- The bindings in the base environment/namespace (currently the same thing) are now locked. This means that the values of base functions cannot be changed except via `assignInNamespace()` and similar tricks.

- `[[` on a factor now returns a one-element factor (and not an integer), `as.list()` on a factor returns a list of one-element factors (and not of character vectors), and `unlist()` on a list of factors returns a factor (and not an integer vector). These changes may affect the results of `sapply()` and `lapply()` applied to factors.

- `mauchly.test()` now returns the W statistic (for comparability with SAS and SPSS), rather than the z (which was accidentally not named in the output)

- `sort(x, decreasing = FALSE, ...)` is now a generic function. This means that `'partial'` is no longer the second argument, and calls which used positional matching may be incorrect: we try to detect them.

- See the section on `'Changes to S4 methods'`: all packages depending on **methods** need to be re-installed.

New features

- `agrep()`, `grep()`, `strwrap()`, `strtrim()`, `substr()` and related functions now coerce arguments which should be character via `as.character()` rather than internally (so method dispatch takes place, e.g. for factors).

  `chartr()`, `charfold()`, `tolower()` and `toupper()` now coerce their main argument if necessary to a character vector via `as.character()`.

  Functions which work element-by-element on character vectors to give a character result now preserve attributes including names, dims and dimnames (as suggested by the Blue Book p. 144). Such functions include `charfold()`, `chartr()`, `gsub()`, `strtrim()`, `sub()`, `substr()`, `tolower()` and `toupper()`. (Note that coercion of a non-character argument may lose the attributes.)

  `agrep(value = TRUE)` preserves names for compatibility with `grep()`.

  `nchar()` has always preserved dims/dimnames (undocumented before) and now also preserves names.

- .Deprecated and .Defunct take a new parameter, msg, that allows for the specification of the message printed and facilitates deprecation of calling sequences etc.

- `.Fortran()` will map `'name'` to lower case, and will work with `'name'` containing underscores.

- The default is now `.saveRDS(compress = TRUE)`

- The `::` operator now also works for packages without name spaces that are on the search path.

- `[[` on a list does not duplicate the extracted element unless necessary. (It did not duplicate in other cases, e.g. a pairlist.)

- `argsAnywhere()` works like `args()` on non-exported functions.

- `as.data.frame()` gains a `'...'` argument.

- Added an `as.data.frame()` method for class `"ftable"`.

- `as.list(<an expression>)` is now handled by internal code and no longer loses attributes such as names.

  `as.list(<a list>)` no longer duplicates (unnecessarily).

- as.POSIX[cl]t can now convert character strings containing fractional seconds.

- `attach()` can now attach a copy of an environment.

- `available.packages()` and `installed.packages()` gain a `'fields'` argument thanks to Seth Falcon.

- `axis.POSIXct()` uses a different algorithm for ranges of 2 to 50 days that will mark days at midnight in the current timezone (even if the graph crosses a DST change).

- `body<-()` and `formals<-()` default to envir = `environment(fun)`, that is they do not by default change the environment. (Previously they changed it to `parent.frame()`.)

- New function `combn(x, m, ..)` for computing on all combinations of size `'m'` (for small `'m'` !).

- The `cumxxx()` functions now handle logical/integer arguments separately from numeric ones, and so return an integer result where appropriate.

- `data.frame()` has a new argument `'stringsAsFactor'`. This and the default for `read.table(as.is=)` are set from the new global option `'stringsAsFactors'` via the utility function `default.stringsAsFactors()`.

- `dev.interactive()` now has an optional argument `'orNone'`.

- `df()` now has a noncentrality argument `'ncp'`, based on a contribution by Peter Ruckdeschel.

- `example()` gains an argument `'ask'` which defaults to `"TRUE when sensible"`, but the default can be overridden by setting option `'example.ask'`.
- `expand.grid()` now has an argument `'KEEP.OUT.ATTRS'` which can suppress (the potentially expensive) `"out.attrs"` attribute. It no longer returns an extraneous `'colnames'` attribute.
- The subset and subassign methods for factors now handle factor matrices, and `dim()` can be set on a factor.
- There is now a `format()` method for class `"ftable"`.
- `head(x, n)` and `tail(x, n)` now also work for negative arguments, thanks to Vincent Goulet.
- `head.matrix()` and `tail.matrix()` are no longer hidden, to be used for building `head()` and `tail()` methods for other classes.
- If `help()` finds multiple help files for a given topic, a menu of titles is used to allow interactive choice.
- `help.search()` now rebuilds the database if `'package'` specifies a package not in the saved database.
- `hist(*, plot = FALSE)` now warns about unused arguments.
- `history()` gains a `'pattern'` argument as suggested by Romain Francois.
- `integer(0)` now prints as that rather than `"numeric(0)"` (it always deparsed as `"integer(0)"`).
- `interaction(..., drop=TRUE)` now gives the same result as `interaction(...)`[,drop=TRUE] (it used to sometimes give a different order for the levels).
- `lag.plot()` produces a conventional plot (not setting mfrow) if only one plot is to be produced.
- `lapply()` does much less copying. Vector `x` are handled without duplication, and other types are coerced via `as.list()`. (As a result, package **boot** runs its examples 4% faster.)

  `lapply(<a pairlist>)` now coerces to a list (rather than traverse the pairlist from the beginning for each item).
- `legend()` has new parameters `'box.lwd'` and `'box.lty'`.
- `lines()` gains a simple method for `isoreg()` results.
- `load()` no longer coerces pairlists to lists (which was undocumented, but has been happening since 1998).
- `make.link()` now returns an object of class `"link-glm"`. The GLM families accept an object of this class for their `'link'` argument, which allows user-specified link functions. Also, `quasi()` allows user-specified variance functions.
- `mapply()` uses names more analogously to `lapply()`, e.g..
- `matplot()` now accepts a `'bg'` argument similarly to `plot.default()` etc.
- `median()` is now generic, and its default method uses `mean()` rather than `sum()` and so is more widely applicable (e.g. to dates).
- Dummy functions `memory.size()` and `memory.limit()` are available on Unix-alikes, for people who have not noticed that documentation is Windows-specific.
- `merge()` works more efficiently when there are relatively few matches between the data frames (for example, for 1-1 matching). The order of the result is changed for `'sort = FALSE'`.
- `merge()` now inserts row names as a character column and not a factor: this makes the default sort order more comprehensible.
- Raw, complex and character vectors are now allowed in model frames (there was a previously undocumented restriction to logical, integer and numeric types.). Character vectors in a formula passed to `model.matrix()` are converted to factors and coded accordingly.
- `modifyList()` utility, typically for housekeeping nested lists.

-
  ```
  x <- 1:20
  y <- rnorm(x)
  nls(y   A*exp(-x^2/sig))
  ```
  no longer returns an unhelpful error message. In this and similar cases, it now tries a wild guess for starting values.
- `Ops.difftime()` now handles unary minus and plus.
- `Ops.Date()` and `Ops.POSIXt()` now allow character arguments (which are coerced to the appropriate class before comparison, for `Ops.POSIXt()` using the current time zone).

- There is a new `option(max.contour.segments = 25000)` which can be raised to allow extremely complex contour lines in `contour()` and `contourLines()`. (PR#9205)

- `options(max.print = N)` where N defaults to 99999 now cuts printing of large objects after about N entries. `print(x, ..., max = N)` does the same for the default method and those building on `print.default()`.

  `options("menu.graphics")` controls if graphical menus should be used when available.

  `options("par.ask.default")` allows the default for `par("ask")` to be set for a newly-opened device. (Defaults to FALSE, the previous behaviour.)

  The way `option("papersize")` is set has been changed. On platforms which support the LC_PAPER locale category, the setting is taken first from the R_PAPERSIZE environment variable at run time, then from the LC_PAPER category (`"letter"` for _US and _CA locales and `"a4"` otherwise). On other platforms (including Windows and older Unixen), the choice is unchanged.

- `package.skeleton()` gains arguments `'namespace'` and `'code_files'`.

- `par(ask=TRUE)` now only applies to interactive R sessions.

- `parse()` now returns up to `'n'` expressions, rather than fill the expressions vector with NULL. (This is now compatible with S.)

- The `'version'` argument for `pdf()` is now increased automatically (with a warning) if features which need a higher level are used.

- `pie()` now allows expressions for `'labels'`, and empty slices.

- There is a new `'%.%'` operator for mathematical annotations (plotmath) which draws a centred multiplication dot (a `\cdot` in LaTeX), thanks to Uwe Ligges.

- `predict.lm()` gains a `'pred.var'` argument. (Wishlist PR#8877.)

- `print.summary.{aov,glm,lm,nls}` and `print.{aov,glm}` make use of `naprint()` to report when na.action altered the model frame.

- `print.table(T, zero.print=ch)` now also replaces 0 by ch when T is non-integer with integer entries.

- Recursive `rapply()` which is similar to lapply but used recursively and can restrict the classes of elements to which it is applied.

- `r2dtable()` has been moved to package **stats**.

- New function `read.DIF()` to read Data Interchange Format files, and (on Windows) this format from the clipboard.

- New experimental function `readNEWS()` to read R's own `"NEWS"` file and similarly formatted ones.

- `readLines()` has a new argument `'warn'` to suppress warnings: the default behaviour is still to warn.

- `reg.finalizer()` has a new argument `'onexit'` to parallel the C-level equivalent R_RegisterFinalizerEx.

- `rep()` is now a primitive function and under some conditions very much faster: `rep.int()` is still a little faster (but does less). (Because it is primitive there are minor changes to the call semantics: see the help page.)

- The `'row.names'` of a data frame may be stored internally as an integer or character vector. This can result in considerably more compact storage (and more logical row names from rbind) when the row.names are `1:nrow(x)`. However, such data frames are not compatible with earlier versions of R: this can be ensured by supplying a character vector as `'row.names'`.

  `row.names()` will always return a character vector, but direct access to the attribute may not.

  The internal storage of row.names = 1:n just records `'n'`, for efficiency with very long vectors.

  The `"row.names"` attribute must be a character or integer vector, and this is now enforced by the C code.

- The `"data.frame"` and `"matrix"` methods for `rowsum()` gain an `'na.rm'` argument.

- Experimental support for memory-use profiling via `Rprof()`, `summaryRprof()`, `Rprofmem()` and `tracemem()`.

- `save.image()` [also called by `sys.save.image()` and hence from `q()`] now defaults to saving compressed binary images. To revert to the previous behaviour set option `"save.image.defaults"`: see `?save.image`.

- There is a new primitive `seq.int()` which is slightly more restricted than `seq()` but often very much faster, and new primitives `seq_along()` and `seq_len()` which are faster still.

- serialize(connection = NULL) now returns a raw vector (and not a character string). unserialize() accepts both old and new formats (and has since 2.3.0).

- setwd() now returns the previously current directory (invisibly).

- The function sort() is now sort.int(), with a new generic function sort() which behaves in the same way (except for the order of its argument list) for objects without a class, and relies on the '[' method for objects with a class (unless a specific method has been written, as it has for class "POSIXlt").

- sort.list() now implements complex vectors (PR#9039), and how complex numbers are sorted is now documented.

- spline() and splinefun() now follow approx[fun] to have an argument 'ties = mean' which makes them applicable also when 'x' has duplicated values.

- str(x) does not print the S3 "class" attribute when it is the same as 'mode' (which is printed anyway, possibly abbreviated) and it puts it beside mode for atomic objects such as S3 class "table".

- str(<data.frame>) now outputs 'data.frame' instead of `data.frame'; this may affect some strict (Package) tests.

- str() now takes also its defaults for 'vec.len' and 'digits.d' from options('str') which can be set by the new strOptions().

- symnum() has a new argument 'numeric.x' particularly useful for handling 0/1 data.

- Sys.getlocale() and Sys.setlocale() support LC_MESSAGES, LC_PAPER and LC_MEASUREMENT if the platform does.

- Sweave has a new options 'pdf.encoding' and 'pdf.version' for its Rweave driver.

- The character vector used by an output textConnection() has a locked binding whilst the connection is open.

  There is a new function textConnectionValue() to retrieve the value of an output textConnection().

- traceback() gains a 'max.lines' argument. .Traceback is no longer stored in the workspace.

- warning(immediate. = TRUE) now applies to getOption("warn") < 0 and not just == 0.

- warnings() is now an accessor function for 'last.warning' (which is no longer stored in the workspace) with a print() method.

- The internal internet download functions have some new features from libxml 2.6.26.

- There is an option "HTTPUserAgent" to set the User Agent in R download requests etc. Patch from S. Falcon.

- PCRE has been updated to version 6.7.

- The C function substituteList now has tail recursion expanded out, so C stack overflow is less likely. (PR#8141, fix by Kevin Hendricks)

- The (somewhat soft) 1023/4 byte limit on command lines is now documented in 'An Introduction to R'.

- The maximum number of open connections has been increased from 50 to 128.

- There is a new manual 'R Internals' on R internal stuctures plus the former appendices of 'Writing R Extensions'.

- The autoloads introduced at the package reorganization have been almost completely removed: the one that remains is for ts().

- The setting of the various Java configuration variables has been improved to refer to JAVA_HOME, and they are now documented in the R-admin manual.

- It is (again) possible to calculate prediction intervals from "lm" objects for the original data frame, now with a warning that the intervals refer to future observations. Weighted intervals have also been implemented, with user-specifiable weights. Warnings are given in cases where the default behaviour might differ from user expectations. See the ?predict.lm for details.

Changes to S4 Methods

- The default prototype object for S4 classes will have its own internal type in 2.4.0, as opposed to being an empty list (the cause of several errors in the code up to 2.3.1). Note that old binary objects, including class definitions, will be inconsistent with the type, and should be recreated.

- S4 method dispatch has been completely revised to use cached generic functions and to search for the best match among inherited methods. See ?Methods and http://developer.r-project.org/howMethodsWork.pdf

- Objects created from an S4 class are now marked by an internal flag, tested by `isS4()` in R and by macro `IS_S4_OBJECT()` in C. This is an efficient and reliable test, and should replace all earlier heuristic tests.
- Some changes have been made to automatic printing of S4 objects, to make this correspond to a call to `show()`, as per `'Programming with Data'`.
- S4 generic and class definitions are now cached when the related package is loaded. This should improve efficiency and also avoid anomalous situations in which a class or generic cannot be found.
- `trace()` now creates a new S4 class for the traced object if required. This allows tracing of user-defined subclasses of `"function"`.

Deprecated & defunct

- The re-named tcltk functions tkcmd, tkfile.tail, tkfile.dir, tkopen, tkclose, tkputs, tkread are now defunct.
- Argument `'col'` of `bxp()` has been removed: use `'boxfill'`.
- Use of NULL as an environment is now an error.
- `postscriptFont()` is defunct: use `Type1Font()`.
- `La.chol()` and `La.chol2inv()` are defunct (they were the same as the default options of `chol()` and chol2inv).
- `La.svd(method = "dgesvd")` is defunct.
- Files install.R and R_PROFILE.R in packages are now ignored (with a warning).
- The following deprecated command-line options to INSTALL have been removed (use the fields in the DESCRIPTION file instead): -s –save –no-save –lazy –no-lazy –lazy-data –no-lazy-data
- Graphical parameter `'tmag'` is obsolete.
- `mauchley.test()` (package **stats**) is now defunct.
- `symbol.C()` and `symbol.For()` are deprecated. They are required in S for use with `is.loaded()`, but are not so required in R.
- `load()`ing an object saved in one of the formats used prior to R 1.4.0 is deprecated. Such objects should be re-saved in the current format.
- `save(version = 1)` is now deprecated.

C-level facilities

- The convenience function ScalarLogical now coerces all non-zero non-NA values to TRUE.

- The vector accessor functions such as INTEGER, REAL and SET_VECTOR_ELT now check that they are called on the correct SEXPTYPE (or at least on a compatible one). See ‘Writing R Extensions’ for the details and for a stricter test regime.

- It is no longer possible to pass list variables to `.C(DUP = FALSE)`: it would have given rise to obscure garbage collection errors.

- allocString is now a macro, so packages using it will need to be reinstalled.

- R_ParseVector was returning with `object(s)` protected in the parser if the status was PARSE_INCOMPLETE or PARSE_ERROR.

- There is a new function Rf_endEmbeddedR to properly terminate a session started by Rf_initEmbeddedR, and both are now available on Windows as well as on Unix-alikes. These and related functions are declared in a new header <Rembedded.h>.

  If R_TempDir is set when embedded R is initialized it is assumed to point to a valid session temporary directory: see ‘Writing R Extensions’.

- There is a new interface allowing one package to make C routines available to C code in other packages. The interface consists of the routines R_RegisterCCallable and R_GetCCallable. These functions are declared in <R_ext/Rdynload.h>. This interface is experimental and subject to change.

  In addition, a package can arrange to make use of header files in another (already installed) package via the `'LinkingTo'` field in the DESCRIPTION file: see `'Writing R Extensions'`.

Utilities

- R CMD SHLIB now handles (as linker commands) -L*, -l* and *.a.

- R CMD check now:

  - warns if there are non-ASCII characters in the R code (as these will likely be syntax errors in some locale).

  - tests Rd cross-references by default, and tests for (syntactically) valid CITATION metadata.

  - tests that the package can be loaded, and that the package and namespace (if there is one) can each be loaded in startup code (before the standard packages are loaded).

  - tests for empty `'exec'` or `'inst'` directories.

  - checks if `$(FLIBS)` is used when `$(BLAS_LIBS)` is.

  - checks that all packages (except non-S4-using standard packages) used in `::`, `:::`, `library()` and `require()` calls are declared in the DESCRIPTION file, and `'methods'` is declared if S4 classes or methods are set.

  - throws an error if the standard packages **methods** and **stats4** are imported from in the NAMESPACE file and not declared in the DESCRIPTION file.

- The test script produced by massage-Examples.pl no longer creates objects in the base environment.

- New utilties R CMD Stangle and R CMD Sweave for extracting S/R code from and processing Sweave documentation, respectively.

- The DESCRIPTION file of packages may contain an `'Enhances:'` field.

- An R CMD javareconf script has been added to allow Java configuration to be updated even after R has been installed.

  INSTALLATION

- The C function realpath (used by `normalizePath()`) is hidden on some systems and we try harder to find it.

- There is a new option `-enable-BLAS-shlib`, which compiles the BLAS into a dynamic library `-lRblas` and links against that. For the pros and cons see the R-admin manual.

  The defaults are now `-without-blas` (so you have explicitly to ask for an external BLAS), and `-enable-BLAS-shlib` unless a usable external BLAS is found or on AIX or on MacOS X 10.2 and earlier.

- MacOS X did not like having LSAME in both BLAS and LAPACK libraries, so it is no longer part of the R-internal LAPACK. We now require an external BLAS to provide LSAME: it seems that nowadays all do.

- The configure test for 'whether mixed C/Fortran code can be run' has been improved as on one system that test passed but the Fortran run-time library was broken.

- A precious configure variable DEFS can be set to pass defines (e.g. -DUSE_TYPE_CHECKING_STRICT) to C code when compiling R.

- There is now a test for visible __libc_stack_end on Linux systems (since it is not visible on some recent glibc's built from the sources).

- MacOS X 10.4 and higher now use two-level namespaces, single module in a shared library and allow undefined symbols to be resolved at run-time. This implies that common symbols are now allowed in package libraries. `-enable-BLAS-shlib` is supported for internal BLAS, external BLAS framework and external static BLAS. An external dynamic library BLAS is NOT supported. (But it can be easily used by replacing internal BLAS library file later.) MacOS X < 10.4 does not support `-enable-BLAS-shlib`.

- Dynamic libraries and modules use a flat namespace on MacOS X 10.4 and higher if either Xcode tools don't support dynamic lookup (Xcode < 2.3) or the FORCE_FLAT_NAMESPACE environment variable is set. (The latter was introduced temporarily for testing purposes and may go away anytime.)

- configure now defaults to `'run-time linking'` on AIX (and AIX < 4.2 is no longer allowed), using -bexpall rather than export/import files. If this works, it allows R to be built in the same way as other Unix-alikes, including with R as a shared library and with a shared BLAS.

- The **mac.binary** package type now defaults to universal binary. If a repository supports architecture-specific Mac binaries, they can be requested by using `"mac.binary.xxx"` in `contrib.url()`, where `xxx` is the desired architecture.

Bug fixes

- The name of a Fortran symbol reported to be missing by `.Fortran()` is now the actual name. (What was reported to be an `'entry point'` was missing the common leading underscore.)

- `print()` on a MBCS character string now works properly a character at a time rather than a byte at time. (This does not affect MBCSs like UTF-8 and the Windows DBCSes which have non-ASCII lead bytes and always worked correctly.)

- `glm()` now recalculates the null deviance whenever there is an offset (even if it is exactly zero to avoid a discontinuity in that case, since the calculations with and without offset are done by different algorithms).

- Amongst families, `quasi()` accepted an expression for link and no other did. Now all accept an expression which evaluates to a one-element character vector (although e.g. `'logit'` is taken as a name and not an expression).

- `trace()` now accepts arguments where= and signature= for the old-style trace (no tracer or exit, edit==FALSE) and just prints a message on entry. Also the undocumented feature of where=function now works for generic functions as well.

- `callNextMethod()` failed for recursive use when the methods had nonstandard argument lists. Now enforces the semantic rule that the inheritance is fixed when the method containing the `callNextMethod()` is installed. See Details in the documentation.

- `UseMethod()` looked for the defining environment of `'generic'` as if it were the current function, although some functions are generic for methods of a different generic.

  Lookup for S3 methods is confined to functions: previously a non-function `'fun.class'` could have masked a function of the same name.

- Line types (lty) specified as hex strings were documented not to allow zero, but some devices accepted zero and handled it in a device-dependent way. Now it is an error on all devices. (PR#8914)

- Subassignment for a time series can no longer extend the series: it used to attempt to but failed to adjust the tsp attributes. Now `window()` must be used.

- Function `AIC()` in package **stats4** was not dispatching correctly on S4 classes via `logLik()` because of namespace issues.

- Subsetting LANGSXPs could break the call-by-value illusion. (PR#7924) (patch from Kevin Hendricks).

- `parse()` with n > 1 gave a syntax error if fewer than n statements were available.

- `parse()` with n > 1 gave strange results on some syntax errors. (PR#8815)

- `lag.plot()` now respects graphical parameters for the axes.

- Using a wrong link in `family()` now gives more consistent error messages.

- `sort.list(method="radix")` works on factors again.

- `object.size()` is more accurate for vector objects (it takes into account the smaller header and also the fixed sizes used in the node classes for small vector objects).

- `addmargins(T, ...)` now returns a `"table"` when `'T'` is a `"table"`, as its help page has always suggested.

- `remove()` now explicitly precludes removing variables from `baseenv()` and throws an error (this was previously ignored).

- Saving the workspace at the end of a session now works as has long been intended, that is it is saved only if something has been added/deleted/changed during the current session.

- The search for bindings in `<<-`, `->>` and `assign(inherits=TRUE)` was omitting the base package, although this was not documented. Now the base package is included (but most bindings there are locked).

- `dweibull(0, shape)` was NaN not Inf for shape < 1. Also, the help for dgamma and dweibull gave support as x > 0, but returned non-zero values for x = 0. (PR#9080)

- Subsetting arrays no longer preserves attributes (it was removed for matrices in 1998).

- The `"factor"` method of `as.character()` no longer maps level `"NA"` to `"<NA>"` (a legacy of before there were NA character strings).

- `terms(keep.order=TRUE)` was not returning a valid `"order"` attribute.

- The DLL registration code was not freeing .External symbols.

- The internet download routines expected URLs of less than 4096 bytes, but did not check. Now this is checked, and http:// URLs are allowed to be up to 40960 bytes.

- `parse(n=-1)` threw a stack-imbalance error, and `parse(n=3)` did not cope correctly with EOF during input.

- Zero-column data frames had no names (rather than `character(0)`).

- `by()` and `acf()` could get confused when they used very long expressions as names.

- `residuals(<glm object>, type="working")` was NA for cases with zero weight (whereas they are well-defined even though the case was not used during the fitting) and the actual value is now returned. This allows residuals to be computed from fits with `'y = FALSE'`.

  The residuals in a fitted `"glm"` object are computed more accurately: the previous formula was subject to cancellation.

- `loess()` now checks the validity of its `'control'` argument.

- `rownames(<0-row matrix>, do.NULL=FALSE)` was wrong. (PR#9136)

- `apply()` now works as documented when applied over 2 or more margins with one of zero extent. (It used to drop dimensions.)

- `head()` and `tail()` now also work row-wise for `"table"` and `"ftable"` objects.

- `NextMethod()` could throw an error/crash if called from a method that was called directly rather than from a generic (so .Method was unset).

- `order(x, na.last = NA)` failed for a zero-length x.

- `grep(pat, x, value = TRUE, perl = L)` preserved names for
  `L == TRUE && !is.na(pat)`
  but not otherwise. Now it always does.

- `[rc]bind()` now find registered methods and not just visible ones.

- Printing a factor no longer ignores attributes such as names and dim/dimnames.

- Command-line arguments after `-encoding` were ignored.

- The check for impossible confidence levels was off by one in wilcox.test (PR#8557)

- `[[` on an environment could create aliases. (PR#8457)

- `pt()` with a very small (or zero) non-centrality parameter could give an unduly stringent warning about `'full precision was not achieved'`. (PR#9171)

- `writeChar()` could segfault if `'nchars'` was given silly values.

- `qt()` and `rt()` did not work for vector `'ncp'`, and `qt()` did not work for negative `'ncp'`.

- `ns()` failed to work correctly when `'x'` was of length one.

- `identical()` ignored tags on pairlists (including names of attributes) and required an identical ordering for attribute values in their pairlists. Now names are compared on pairlists, and attribute sets are treated as unordered.

- If they were unused arguments supplied to a closure, only the first non-empty one was reported, despite the message.

  Unmatched empty arguments (such as `f(1,,)` for a function of one argument) were ignored. They are now an error.

- Calling a builtin with empty arguments used to silently remove them (and this was undocumented). Now this is an error unless builtin is `c()` or `list()` or there are only trailing empty arguments, when it is a warning (for the time being: this will be made an error in R 2.5.0).

- `install.packages()` ignored `'configure.args'` if the vector was unnamed.

- `biplot()` now works if there are missing values in the data.

- `biplot()` now passes `par()` values to all four axes (not just those on sides 1 and 2).

- `[.acf` now handles an empty first index.

- Deparsing uses backticks more consistently to quote non-syntactic names.

- Assigning to the symbol in a `for()` loop with a list/expression/pairlist index could alter the index. Now the loop variable is explicitly read-only. (PR#9216)

- Using `old.packages()` (and hence `update.packages()`) on an empty (or non-existent) library failed with an obscure message.

- `plot.xy()` could segfault if supplied with an invalid `'col'` argument. (PR#9221)

- `menu()` with graphics=TRUE attempted to use Tcl/Tk on unix even if DISPLAY was not set (in which case Tk is not available and so the attempt is bound to fail).

- The `print()` method for `'dist'` objects prints a matrix even for n = 2.

- The `cum<xxx>` functions were missing some PROTECTs and so could segfault on long vectors (especially with names or where coercion to numeric occurred).

- The `X11()` device no longer produces (apparently spurious) 'BadWindow (invalid Window parameter)' warnings when run from Rcmdr.

- `legend()` assumed that widths and heights of strings were positive, which they need not be in user coordinates with reversed axes. (In part, PR#9236)

- The `plot()` methods for `"profile.nls"` objects could get confused if `'which'` had been used in the `profile()` call. (PR#9231)

- `boxplot()` did not passed named arguments (except graphics parameters) to `bxp()` as documented. (PR#9183)

- Only genuinely empty statements act as 'return' in the browser, not say those starting with a comment char. (PR#9063)

- `summary.mlm()` incorrectly used accessor functions to fake an `"lm"` object. (PR#9191)

- `prettyNum()` was not preserving attributes, despite being explicitly documented to. (PR#8695)

- It was previously undocumented what happened if a graphical parameter was passed in both `'...'` and `'pars'` to `boxplot()` and `bxp()`, and they behaved differently. Now those passed in `'...'` have precedence in both cases.

- A failed subassignment could leave behind an object `'*tmp*'`. The fix also sometimes gives better error messages.

- Using SIGUSR1 on Unix now always terminates a session, and no longer is caught by browser contexts and restarts (such as `try()`).

- In the **graphics** package, in-line `'font=5'` was being ignored (report by Tom Cook).

- `nls()` looked for non-parameter arguments in a function call in the wrong scope (from the body of nls).

- Printing of complex numbers could misbehave when one of the parts was large (so scientific notation was used) and the other was so much smaller that it had no significant digits and should have been printed as zero (e.g. 1e80+3e44i).

- Using install.packages with type=`"mac.binary"` and target path starting with failed with a cryptic message while unpacking.

- `getwd()` now works correctly when the working directory is unavailable (e.g. unreadable).

- The alternative hypothesis in `wilcox.test()` was labelled by an unexplained quantity `'mu'` which is now spelled out.

  The alternative hypothesis in `ks.test()` is clearer both in the documentation and in the result. (PR#5360)

Changes on CRAN

by Kurt Hornik

New contributed packages

AdaptFit Adaptive semiparametric regression. Fits semiparametric regression models with spatially adaptive penalized splines. By Tatyana Krivobokova.

CompetingRiskFrailty Competing risk model with frailties for right censored survival data. Offers fitting of smooth varying coefficients in a competing risks modeling of hazards as well as estimating of the frailties (or unobserved heterogeneities) for clustered observations. Nonparametric penalized spline (p-spline) fitting of smooth covariates effects is proposed. As a spline basis truncated polynomial functions are chosen. The frailties are also fitted (via the EM algorithm) in a flexible way using a penalized mixture of Gamma distributions. By Pavel Khomski.

DPpackage Semiparametric Bayesian analysis using Dirichlet process priors. Contains functions for posterior analysis for a number of semiparametric statistical models. Simulation is done in compiled Fortran. By Alejandro Jara

Vallejos.

FLEDA Exploratory Data Analysis for **FLR**. Develops several procedures to explore fisheries data. By Ernesto Jardim and Manuela Azevedo.

Flury Data sets from Bernard Flury (1997), "A First Course in Multivariate Statistics", Springer NY. By Bernard Flury.

GAMBoost Routines for fitting generalized additive models by likelihood based, using penalized B-splines. By Harald Binder.

GeneCycle Identification of periodically expressed genes. Implements the approaches of Wichert et al. (2004) and Ahdesmaki et al. (2005) for detecting periodically expressed genes from gene expression time series data. By Miika Ahdesmaki, Konstantinos Fokianos, and Korbinian Strimmer.

GeneNet Modeling and inferring gene networks. A package for analyzing gene expression (time series) data with focus on the inference of gene networks. In particular, it implements the methods of Schaefer and Strimmer (2005a,b,c) and Opgen-Rhein and Strimmer (2006) for learning large-scale gene association networks. For plotting the gene networks, **GeneNet** uses the **graph** and **Rgraphviz** packages, available from http://www.bioconductor.org. By Rainer Opgen-Rhein, Juliane Schaefer, and Korbinian Strimmer.

HH Support software for the book "Statistical Analysis and Data Display" by Richard M. Heiberger and Burt Holland (Springer, ISBN 0-387-40270-5). This contemporary presentation of statistical methods features extensive use of graphical displays for exploring data and for displaying the analysis. The authors demonstrate how to analyze data—showing code, graphics, and accompanying computer listings—for all the methods they cover. They emphasize how to construct and interpret graphs, discuss principles of graphical design, and show how accompanying traditional tabular results are used to confirm the visual impressions derived directly from the graphs. Many of the graphical formats are novel and appear for the first time in print. By Richard M. Heiberger.

JGR JGR — Java Gui for R. By Markus Helbig.

JavaGD Java Graphics Device, routing all graphics commands to a Java program. The actual functionality of the JavaGD depends on the Java-side implementation. Simple AWT and Swing implementations are included. By Simon Urbanek.

MBA Multilevel B-spline Approximation for scattered data. By Andrew O. Finley and Sudipto Banerjee.

MBESS Methods for the Behavioral, Educational, and Social Sciences. Implements methods that are especially useful to researchers working within these sciences (both substantive researchers and methodologists). Many of the methods contained within **MBESS** are applicable to quantitative research in general. By Ken Kelley.

MChtest Monte Carlo hypothesis tests. Allows a couple of different sequential stopping boundaries (a truncated sequential probability ratio test boundary and a boundary proposed by Besag and Clifford, 1991). Gives valid p-values and confidence intervals on p-values. By M. P. Fay.

MFDA Model based Functional Data Analysis, for doing model based functional clustering. By Wenxuan Zhong and Ping Ma.

MKLE Maximum Kernel Likelihood Estimation. By Thomas Jaki.

MasterBayes Maximum likelihood and Markov Chain Monte Carlo methods for pedigree reconstruction, analysis and simulation. The primary aim is to use MCMC techniques to integrate over uncertainty in pedigree configurations estimated from molecular markers and phenotypic data. Emphasis is put on the marginal distribution of parameters that relate the phenotypic data to the pedigree. All simulation is done in compiled C++ for efficiency. By Jarrod Hadfield.

PBSmodelling Provides software to facilitate the design, testing, and operation of computer models. It focuses particularly on tools that make it easy to construct and edit a customized graphical user interface (GUI). Although it depends heavily on the R interface to the Tcl/Tk package, a user does not need to know Tcl/Tk. The package contains examples that illustrate models built with other R packages, including PBS Mapping, **odesolve**, and **BRugs**. It also serves as a convenient prototype for building new R packages, along with instructions and batch files to facilitate that process. By Jon T. Schnute, Alex Couture-Beil, and Rowan Haigh.

PearsonICA Independent component analysis using score functions from the Pearson system. The Pearson-ICA algorithm is a mutual information-based method for blind separation of statistically independent source signals. It has been shown that the minimization of mutual information leads to iterative use of score functions, i.e., derivatives of log densities. The Pearson system allows adaptive modeling of score functions. The flexibility of the Pearson system makes it possible to model a wide range of source distributions including asymmetric distributions. The algorithm is designed especially for problems with asymmetric sources but it works for symmetric sources as well. By Juha Karvanen.

ProbeR Reliability for gene expression from Affymetrix chips. Most statistical methods for finding interesting genes are focusing on the summary value with large fold change or large variations. Very few methods consider the probe level data. This package is to calculate the reliability of the gene expression data from Affymetrix chips using the probe-level data. By Eun-Kyung Lee.

QCAGUI A graphical user interface (GUI) for the **QCA** package, derived from R Commander. Because QCA has little to do with statistics, the menus from **Rcmdr** were stripped down to the very basics. In crisp sets QCA data is binary therefore it is fairly decent to treat it as categorical (1: presence; 0: absence). In order to ease the primary analysis (e.g., tables of frequencies) and the creation of basic graphs, this package activates some menus that are not available in **Rcmdr** but for factors. Users should be aware, however, that **QCAGUI** is *not* a package for statistics; **Rcmdr** is better for this purpose. This package is 99% based on the **Rcmdr** package written by John Fox. Only two additional menus were adapted by Adrian Dusa.

R.rsp R Server Pages. An R Server Page (RSP) is a document that contains both text in a format of choice (HTML, $\TeX$, ...) as well as R source code within special tags. An RSP file can be translated into a so called R servlet, which is an R script that outputs the final document when sourced. This way documents in any format can be generated dynamically using R, e.g., automatic reports of statistical analysis. Utilizing an internal cross-platform web server, this package provides dynamic help pages in HTML. If other packages provide RSP help pages, these are automatically linked to in the RSP main menu. By Henrik Bengtsson.

RGtk2 R bindings for the Gimp Tool Kit (Gtk) 2.0. By Duncan Temple Lang and Michael Lawrence.

RJaCGH Reversible Jump MCMC for the analysis of CGH arrays: Bayesian analysis of CGH microarrays fitting Hidden Markov Chain models. The selection of the number of states is made via their posterior probability computed by Reversible Jump Markov Chain Monte Carlo Methods. By Oscar Rueda and Ramon Diaz-Uriarte.

RTisean R interface to TISEAN algorithms. Algorithms for time series analysis from nonlinear dynamical systems theory originally made available by Rainer Hegger, Holger Kantz and Thomas Schreiber at `http://www.mpipks-dresden.mpg.de/~tisean/`. A related R package (**tseriesChaos** by Antonio, Fabio Di Narzo) contains rewritten versions of a few of the TISEAN algorithms. The intention of the present package is to use the TISEAN routines from within R with no need of manual importing/exporting. This package only contains R interface code, and requires that you have the TISEAN algorithms available on your computer. R interface code by Antonio Fabio Di Narzo, R documentation and examples by Gianluca Gazzola.

SNPassoc SNPs-based whole genome association studies. This package carries out most common analyses when performing whole genome association studies. These analyses include descriptive statistics and exploratory analysis of missing values, calculation of Hardy-Weinberg equilibrium, analysis of association based on generalized linear models (either for quantitative or binary traits), and analysis of multiple SNPs (haplotype and epistasis analysis). By Juan R González, Lluís Armengol, Elisabet Guinó, Xavier Solé, and Víctor Moreno.

SQLiteDF Stores data frames and matrices in SQLite tables. By Miguel A. R. Manese.

TSP Traveling Salesperson Problem (TSP). Basic infrastructure and some algorithms for the traveling salesperson problem (TSP). The package provides some simple algorithms and an interface to Concorde, the currently fastest TSP solver. The solver itself is not included in the package and has to be obtained separately. Note that the package currently only implements support for the symmetric TSP. By Michael Hahsler and Kurt Hornik.

TWIX Trees WIth eXtra splits. By Sergej Potapov.

TwoWaySurvival Additive two-way hazards modeling of right censored survival data. Offers fitting of smooth varying coefficients in an additive two-way hazard model. Nonparametric penalized spline (p-spline) fitting is proposed. As a spline basis truncated polynomial functions are chosen. By Pavel Khomski.

ada Performs discrete, real, and gentle boost under both exponential and logistic loss on a given data set. Provides a straightforward, well-documented, and broad boosting routine for classification, ideally suited for small to moderate-sized data sets. By Mark Culp, Kjell Johnson, and George Michailidis.

adabag Implements Freund and Schapire's Adaboost.M1 algorithm and Breiman's Bagging algorithm using classification trees as individual classifiers. Once these classifiers have been trained, they can be used to predict on new data. Also, cross validation predictions can be done. By Esteban Alfaro Cortés, Matías Gámez Martínez and Noelia García Rubio.

ade4TkGUI A Tcl/Tk GUI for some basic functions in the **ade4** package. By Jean Thioulouse and Stephane Dray.

agsemisc Miscellaneous plotting and utility functions. High-featured panel functions for `bwplot` and `xyplot`, various plot management helpers, some other utility functions. By Lutz Prechelt.

allelic A fast, unbiased and exact allelic exact test. This is the implementation in R and C of a new association test described in "A fast, unbiased and exact allelic exact test for case-control association studies" (submitted). It appears that in most cases the classical chi-square test used for testing for allelic association on genotype data is biased. Our test is unbiased, exact but fast through careful optimization. By Karl Forner.

aspace A collection of functions for computing centrographic statistics (e.g., standard distance, standard deviation ellipse), and minimum convex polygons (MCPs) for observations taken at point locations. A tool is also provided for converting geometric objects associated with the centrographic statistics, and MCPs into ESRI shape files. By Tarmo K. Remmel and Ron N. Buliung.

backtest Exploring portfolio-based hypotheses about financial instruments (stocks, bonds, swaps, options, et cetera). By Kyle Campbell, Jeff Enos, Daniel Gerlanc, and David Kane.

biglm Bounded memory linear and generalized linear models: regression for data too large to fit in memory. By Thomas Lumley.

binom Binomial confidence intervals for several parametrizations. By Sundar Dorai-Raj.

cacheSweave Tools for caching Sweave computations and storing them in filehash databases. By Roger D. Peng.

cairoDevice Loadable Cairo/GTK device driver for R which, in addition to standalone functionality, can be used to create devices as embedded components in a GUI using a Gtk drawing area widget, e.g., using **RGtk**. By Michael Lawrence and Lyndon Drake.

chemCal Calibration functions for analytical chemistry. Provides simple functions for plotting linear calibration functions and estimating standard errors for measurements according to the Handbook of Chemometrics and Qualimetrics: Part A by Massart et al. There are also functions estimating the limit of detection (LOQ) and limit of quantification (LOD). The functions work on model objects from (optionally weighted) linear regression (`lm`) or robust linear regression (`rlm` from the **MASS** package). By Johannes Ranke.

compOverlapCorr Comparing overlapping correlation coefficients. Contains a function to test the difference between two overlapping (in the sense of having a variable in common) correlation coefficients, using a *Z*-test as described by Meng, Rosenthal, and Rubin (1992). By Ka-Loh Li and Xiaoping Zhu.

connectedness Functions to find, plot and subset disconnected sets in a two-way classification without interaction. By Gregor Gorjanc (colors were taken from **RColorBrewer**).

cts Continuous time autoregressive models and the Kalman filter. Fortran original by G. Tunnicliffe-Wilson and Zhu Wang, R port by Zhu Wang.

delt Estimation of multivariate densities with adaptive histograms (greedy histograms and CART histograms), stagewise minimization, and bootstrap aggregation. By Jussi KlemelÃd'.

distributions Probability distributions (binomial, poisson, geometric, normal, chi-squared, Fisher, Student) based on TI-83 Plus graphic scientific calculator. By Fabio Frascati.

dlm Maximum likelihood and Bayesian analysis of dynamic linear models. By Giovanni Petris.

eRm Estimating extended Rasch models. Allows the estimation of Rasch models and extensions such as: Linear logistic test model (LLTM), rating scale model (RSM), linear rating scale model (LRSM), partial credit model (PCM), and linear partial credit model (LPCM). By Patrick Mair and Reinhold Hatzinger.

eiPack Ecological inference and higher-dimension data management. Provides methods for analyzing $R \times C$ ecological contingency tables using the extreme case analysis, ecological regression, and Multinomial-Dirichlet ecological inference models. Also provides tools for manipulating higher-dimensional data objects. By Olivia Lau, Ryan T. Moore, and Michael Kellermann.

exactmaxsel Computes the exact distribution of some maximally selected statistics in the following setting: the "response" variable is binary, the splitting variable may be nominal, ordinal or continuous. Currently, the package implements the chi-squared statistic and the Gini index. By Anne-Laure Boulesteix.

fdrtool Contains utility functions for controlling and estimating (local) false discovery rates, including some diagnostic plots to support the choice of the most suitable fdr method. By Korbinian Strimmer.

feature Feature significance for multivariate kernel density estimation. By Tarn Duong and Matt Wand.

ffmanova Fifty-fifty MANOVA. Performs general linear modeling with multiple responses (MANCOVA). An overall *p*-value for each model term is calculated by the 50-50 MANOVA method, which handles collinear responses. Rotation testing is used to compute adjusted single response *p*-values according to familywise error rates and false discovery rates. By ÃŸyvind Langsrud and BjÃÿrn-Helge Mevik.

fingerprint Functions to manipulate binary fingerprints of arbitrary length. A fingerprint is represented as a vector of integers, such that each element represents the position in the fingerprint that is set to 1. Basic operations such as logical AND, OR, NOT and XOR are supplied. Distance metrics are also available. Fingerprints can be converted to Euclidean vectors (i.e., points on the unit hypersphere) and can also be folded using XOR. Then, arbitrary fingerprint formats can be handled via line handlers. By Rajarshi Guha.

fmri Analysis of fMRI experiments as described in Tabelow, Polzehl, Spokoiny, WIAS-preprint No. 1079, 2005. By Karsten Tabelow and Joerg Polzehl.

forecasting Functions and data sets for forecasting. By Rob J Hyndman.

gRcox Estimation, model selection and other aspects of statistical inference in Graphical Gaussian models with edge and vertex symmetries (colored Graphical Gaussian models). By SÃÿren HÃÿjsgaard and Steffen L. Lauritzen.

gWidgets Provides a toolkit-independent API for building interactive GUIs. By Philippe Grosjean, Michael Lawrence, Simon Urbanek, and John Verzani.

gWidgetsRGtk2 Toolkit implementation of **gWidgets** for **RGtk2**. By Michael Lawrence and John Verzani.

gamlss.nl A GAMLSS add on package for fitting non linear parametric models. The main function `nlgamlss()` can fit any parametric (up to four parameter) GAMLSS distribution. By Mikis Stasinopoulos and Bob Rigby, with contributions from Philippe Lambert.

gamlss.tr A GAMLSS add on package for generating and fitting truncated (gamlss.family) distributions. The main function `gen.trun()` generates truncated version of an existing GAMLSS family distribution. By Mikis Stasinopoulos and Bob Rigby.

gbev Gradient boosted regression trees with errors-in-variables. Performs non-parametric regression when covariates are measured with error. The models are estimated using gradient boosted regression trees. Regression is performed using squared error loss, while binary response regression can be performed using negative log-likelihood loss. By Joe Sexton.

grImport Functions for converting, importing, and drawing PostScript pictures in R plots. By Paul Murrell and Richard Walton.

grplasso Fits user specified models with group lasso penalty. By Lukas Meier.

gvlma Global validation of linear models assumptions, providing methods from the paper "Global Validation of Linear Model Assumptions" by E. A. Pena and E. H. Slate (JASA, 101(473):341-354, 2006). By Edsel A. Pena and Elizabeth H. Slate.

hot Computation on micro-arrays. By Gilles Guillot.

ifa Independent factor analysis. By Cinzia Viroli.

iplots Interactive plots for R. By Simon Urbanek and Tobias Wichtrey.

laser Likelihood Analysis of speciation/extinction rates from phylogenies. Implements maximum likelihood methods based on the birth-death process to test whether diversification rates have changed over time. Permits batch processing of phylogenies to generate null distributions of test statistics and posterior distributions of parameter estimates. Additional functions for manipulating branching times from molecular phylogenies and for simulating branching times under constant-rate models of diversification are provided. By Dan Rabosky.

lawstat Statistical tests widely utilized in biostatistics, public policy and law. Along with the well known tests for equality of means and variances, randomness, measures of relative variability, etc., this package contains new robust tests of symmetry, omnibus and directional tests of normality, and their graphical counterparts such as Robust QQ plot; a robust trend test for variances, etc. All implemented tests and methods are illustrated by simulations and real-life examples from legal statistics, economics and biostatistics. By Wallace Hui, Yulia R. Gel, Joseph L. Gastwirth, and Weiwen Miao.

ldbounds Lan-DeMets method for group sequential boundaries. By Charlie Casper and Oscar A. Perez.

lhs Provides a number of methods for creating and augmenting Latin Hypercube Samples. By Rob Carnell.

logcondens Estimate a log-concave probability density from i.i.d. observations. Given independent and identically distributed observations $X(1), \ldots, X(n)$, this package allows to compute a concave, piecewise linear function ϕ on $[X(1), X(n)]$ with knots only in $\{X(1), X(2), \ldots, X(n)\}$ such that $L(\phi) = \sum_{i=1}^{n} W(i) * \phi(X(i)) - \int_{X(1)}^{X(n)} \exp(\phi(x))\,dx$ is maximal, for some weights $W(1), \ldots, W(n)$ such that $\sum_{i=1}^{n} W(i) = 1$. According to the results in Duembgen and Rufibach (2006), this function ϕ maximizes the ordinary log-likelihood $\sum_{i=1}^{n} W(i) * \phi(X(i))$ under the constraint that ϕ is concave. The corresponding function $\exp(\phi)$ is a log-concave probability density. Two algorithms are offered: An active set algorithm and one based on the pool-adjacent-violators algorithm. By Kaspar Rufibach and Lutz Duembgen.

lvplot Implements letter value boxplots which extend the standard boxplot to deal with larger data. By Heike Hofmann.

mboost Model-based boosting. Functional gradient descent algorithms (boosting) for optimizing general loss functions utilizing componentwise least squares, either of parametric linear form or smoothing splines, or regression trees as base learners for fitting generalized linear, additive and interaction models to potentially high-dimensional data. By Torsten Hothorn and Peter Bühlmann.

mclust02 Model-based cluster analysis: the 2002 version of MCLUST. By C. Fraley and A. E. Raftery; R port by Ron Wehrens.

mprobit Multivariate probit model for binary/ordinal response. Multivariate normal rectangle probabilities (positive exchangeable, general, approximations); MLE of regression and correlation parameters in the multivariate binary/ordinal probit models: exchangeable, AR(1), and unstructured correlation matrix. By Harry Joe, Laing Wei Chou, and Hongbin Zhang.

mratios Inferences for ratios of coefficients in the general linear model. In particular, tests and confidence interval estimations for ratios of treatment means in the normal one-way layout and confidence interval estimations like in (multiple) slope ratio and parallel line assays can be carried out. Moreover, it is possible to calculate the sample sizes required in comparisons with a control based on relative margins. For the simple two-sample problem, functions for a t-test for ratio-formatted hypotheses and the corresponding Fieller confidence interval are provided assuming homogeneous or heterogeneous group variances. By Gemechis Dilba and Frank Schaarschmidt.

multcompView Visualizations of paired comparisons. Convert a logical vector or a vector of p-values or a correlation, difference, or distance matrix into a display identifying the pairs for which the differences were not significantly different. Designed for use in conjunction with the output of functions like `TukeyHSD`, `dist`, `simint`, `simtest`, `csimint`, `csimtest` (from **multcomp**), `friedmanmc` and `kruskalmc` (from **pgirmess**). By Spencer Graves and Hans-Peter Piepho, with help from Sundar Dorai-Raj.

multic Calculate the polygenic and major gene models for quantitative trait linkage analysis using variance components approach. By Eric Lunde, Mariza de Andrade and Beth Atkinson.

multilevel Functions for the analysis of multilevel data by organizational and social psychologists. Includes a number of estimates of within-group agreement, and a series of routines using random group resampling (RGR) to identify the group-level properties of data. Finally, the package contains some basic simple routines for estimating reliability. By Paul Bliese.

mvtnormpcs Multivariate normal and T Distribution functions of Dunnett (1989). Computes multivariate student and multivariate normal integrals, given a correlation matrix structure defined by a vector `bpd`, such that $\rho(i, j) = bpd(i) * bpd(j)$ (product correlation structure). By Duane Currie and Jianan Peng, using code from Dunnett (Applied Statistics, 1989).

nFactors Non graphical solution to the Cattell Scree test, based on using acceleration factors or optimal coordinates (with or without Parallel Analysis) as numerical indices. By Gilles Raiche.

odfWeave Sweave processing of Open Document Format (ODF) files. By Max Kuhn.

paleoTS Modeling evolution in paleontological time-series. Facilitates analysis of paleontological sequences of trait values from an evolving lineage. Functions are provided to fit, using maximum likelihood, evolutionary models including random walk (biased and unbiased forms) and stasis models. By Gene Hunt.

partitions Additive partitions of integers. Enumerates the partitions, unequal partitions, and restricted partitions of an integer; the three corresponding partition functions are also given. By Robin K. S. Hankin.

pcalg Standard and robust estimation of the skeleton (ugraph) of a Directed Acyclic Graph (DAG) via the PC algorithm. By Markus Kalisch and Marin Maechler.

plm A set of estimators and tests for panel data. By Yves Croissant.

pmg Poor Man's GUI: simple GUI for R using **gWidgets**. By John Verzani.

poLCA Polytomous variable Latent Class Analysis: latent class analysis and latent class regression models for polytomous outcome variables. Also known as latent structure analysis. By Drew Linzer and Jeffrey Lewis.

powell Optimizes a function using Powell's UObyQA algorithm. By Sundar Dorai-Raj, original Fortran from Mike Powell.

prettyR Functions for conventionally formatted descriptive stats. By Jim Lemon and Phillipe Grosjean.

proptest Tests of the proportional hazards assumption in the Cox model: data-driven Neyman type smooth tests and score process based tests for identifying nonproportional covariates. By David Kraus.

psychometric Applied psychometric theory. Contains functions useful for correlation theory, meta-analysis (validity-generalization), reliability, item analysis, inter-rater reliability, and classical utility. By Thomas D. Fletcher.

qp q-order partial correlation graph search (q-partial or, for short, qp) algorithm, a robust procedure for structure learning of undirected Gaussian graphical Markov models from "small n, large p" data, that is, multivariate normal data coming from a number of random variables p larger than the number of multidimensional data points n as in the case of, e.g., microarray data. By Robert Castelo and Alberto Roverato.

qtlbim QTL Bayesian Interval Mapping: Functions for model selection for genetic architecture. By Brian S. Yandell and Nengjun Yi, with contributions from Tapan Mehta, Samprit Banerjee and Daniel Shriner (UA-Birmingham), W. Whipple Neely (UW-Madison) and Hao Wu and Randy von Smith (Jackson Laboratory).

random An interface to the true random number service provided by the random.org website created by Mads Haahr. The random.org web service samples atmospheric noise via radio tuned to an unused broadcasting frequency together with a skew correction algorithm due to John von Neumann. More background is available in the included vignette based on an essay by Mads Haahr. In its current form, the package offers functions to retrieve random integer number (with duplicates), randomized sequences (without duplicates) and raw random bytes. By Dirk Eddelbuettel.

randomSurvivalForest Ensemble survival analysis based on a random forest of trees using random inputs. By Hemant Ishwaran and Udaya B. Kogalur.

rattle A graphical user interface for data mining in R using GTK. Provides a Gnome (**RGtk2**) based interface to R functionality for data mining. The aim is to provide a simple and intuitive interface that allows a user to quickly load data from a CSV file, explore the data, build some models, and evaluate those models, knowing little about R. All R commands are logged and available for the user, as a tool to then begin interacting directly with R itself, if so desired. By Graham Williams.

reweight Adjustment of survey respondent weights. Adjusts the weights of survey respondents so that the marginal distributions of certain variables fit more closely to those from a more precise source (e.g., Census Bureau's data). By Feiming Chen.

rhosp Side effect risks in hospital: simulation and estimation. Evaluating risk (that a patient arises a side effect) during hospitalization is the main purpose of this package. Several methods (parametric, non parametric and De Vielder estimation) to estimate the risk constant (R) are implemented in this package. There are also functions to simulate the different models of this issue in order to quantify the previous estimators. By Christophe Dutang and Julie BarthÃl's.

robust A package of robust methods from Insightful. By Jeff Wang, Ruben Zamar, Alfio Marazzi, Victor Yohai, Matias Salibian-Barrera, Ricardo Maronna, Eric Zivot, David Rocke, Doug Martin, and Kjell Konis.

rpanel Provides a set of functions to build simple GUI controls for R functions, using the **tcltk** package. Uses could include changing a parameter on a graph by animating it with a slider or a "doublebutton", up to more sophisticated control panels. By Bowman, Bowman and Crawford.

rpubchem Interface to the PubChem Collection. Access PubChem data (compounds, substance, assays). Structural information is provided in the form of SMILES strings. This package only provides access to a subset of the precalculated data stored by PubChem. Bio-assay data can be accessed to obtain descriptions as well as the actual data. It is also possible to search for assay IDs by keyword. Currently the main limitation is that only 1000 molecules can be downloaded at a time from the PubChem servers. By Rajarshi Guha.

scaleboot Approximately unbiased p-values via multiscale bootstrap. By Hidetoshi Shimodaira.

sensitivity Sensitivity Analysis. Implemented methods are: linear sensitivity analysis (SRC, PCC, ...), the screening method of Morris and non-linear global sensitivity analysis (Sobol indices, FAST method). By Gilles Pujol (this package was originally developed at Commissariat a l'Energie Atomique CEA, Service d'Etudes et de Simulation du Comportement des Combustibles CEA/DEN/CAD/DEC/SESC, France).

sigma2tools Test of hypothesis about σ^2. By Fabio Frascati.

simecol An object oriented framework and tools for SIMulation of ECOLogical (and other) dynamic systems. By Thomas Petzoldt.

spBayes Fits Gaussian models with potentially complex hierarchical error structures using Markov chain Monte Carlo (MCMC). By Andrew O. Finley, Sudipto Banerjee, and Bradley P. Carlin.

spatclus Arbitrarily shaped multiple spatial cluster detection for 2D and 3D spatial point patterns (case event data). The methodology of this package is based on an original method that allows the detection of multiple clusters of any shape. A selection order and the distance from its nearest neighbor once pre-selected points have been taken into account are attributed at each point. This distance is weighted by the expected distance under the uniform distribution hypothesis. Potential clusters are located by modeling the multiple structural change of the distances on the selection order. Their presence is tested using the double maximum test and a Monte Carlo procedure. By Christophe Demattei.

spatialkernel Edge-corrected kernel density estimation and binary kernel regression estimation for multivariate spatial point process data. By Pingping Zheng and Peter Diggle.

spgrass6 Interface between GRASS 6.0 geographical information system and R. By Roger Bivand.

st Implements the "shrinkage t" statistics described in Opgen-Rhein and Strimmer (2006). Also offers a convenient interface to a number of other regularized t-type statistics often used in high-dimensional case-control studies. By Rainer Opgen-Rhein and Korbinian Strimmer.

startupmsg Utilities for start-up messages. By Peter Ruckdeschel.

stepPlr L_2 penalized logistic regression for both continuous and discrete predictors, with the forward stepwise variable selection procedure. By Mee Young Park and Trevor Hastie.

stinepack Interpolation routine using the Stineman algorithm. By Tomas Johannesson and Halldor Bjornsson.

svcR Implements a support vector machine technique for clustering. By Nicolas Turenne.

tensorA Provides convenience functions for advanced linear algebra with tensors and computation with data sets of tensors on a higher level abstraction. Includes Einstein and Riemann summing conventions, dragging, co- and contravariate indices, and parallel computations on sequences of tensors. By K. Gerald van den Boogaart.

titecrm Time-to-event continual reassessment method and calibration tools. Provides functions to run the TITE-CRM in phase I trials and the calibration tools to plan a TITE-CRM design. By Ken Cheung.

triangle Provides the standard distribution functions for the triangle distribution. By Rob Carnell.

trimcluster Trimmed *k*-means clustering. The package is planned to be expanded to contain further trimmed clustering methods developed at the University of Valladolid. By Christian Hennig.

tsDyn Time series analysis based on dynamical systems theory. By Antonio Fabio Di Narzo.

tutoR Mask common functions so that bad inputs are picked up in advance. Errors are explained prior to or instead of function execution. `eg()` picks out "Examples" first and foremost. In `deskcheck()` the debug flag is set and execution initiated for a function to be tested. By Mark Fielding.

unbalhaar Implements an algorithm for nonparametric function estimation using Unbalanced Haar wavelets. By Piotr Fryzlewicz.

vars Estimation, lag selection, diagnostic testing, forecasting, causality analysis, forecast error variance decomposition and impulse response functions of VAR models and estimation of SVAR models (A-model, B-model, AB-model). By Bernhard Pfaff.

wnominate Estimates Poole and Rosenthal W-NOMINATE scores from roll call votes supplied though a roll call object from package **pscl**. By Keith Poole, Jeffrey Lewis, James Lo, and Royce Carroll.

xlsReadWrite Natively read and write Excel (v97-2003/BIFF8) files. By Hans-Peter Suter.

zipfR Statistical models and utilities for the analysis of word frequency distributions. The utilities include functions for loading, manipulating and visualizing word frequency data and vocabulary growth curves. The package also implements several statistical models for the distribution of word frequencies in a population. (The name of this package derives from the most famous word frequency distribution, Zipf's law.). By Stefan Evert and Marco Baroni.

Other changes

- Packages **data.table** and **pls.pcr** were moved from the main CRAN section to the Archive.

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

R Foundation News

by Kurt Hornik

Donations and new members

New supporting institutions

Massachusetts General Hospital Biostatistics Center, Boston, USA

National Public Health Institute, Helsinki, Finland

New supporting members

Micah Altman (USA)
Robert M. Baskin (USA)
Annette Baumann (Germany)
Abdu Benheddi (Switzerland)
David Booth (USA)
Rafael Borges (USA)
Lance Brannmann (USA)
Romain Francois (France)
John Gavin (UK)
David Henderson (USA)
Paulo Justiniano (Brasil)
Stephen Kaluzny (USA)
John Maindonald (Australia)
Masafumi Okada (Japan)
Friedrich Schuster (Germany)
Klemens Vierlinger (Austria)

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

News from the Bioconductor Project

by Seth Falcon

Bioconductor 1.9 was released on October 4, 2006 and is designed to be compatible with R 2.4.0, released one day before Bioconductor. There are 26 new packages in the release and many improvements have been made to existing packages.

One area that has flourished in this release is software for analyzing cell-based assays. Below we summarize these developments. Of particular interest to the R community may be the **EBImage** package which provides an interface to the ImageMagick image processing library.

cellHTS Provides a comprehensive set of tools for the preprocessing, quality assessment and visualization of data from systematic cell-based screens, for example RNAi screens, and for the detection of effectors.

prada, **rflowcyt** Both provide import and management of data from flow cytometry, many visualization functions, and tools that help to do regression modeling on these data.

EBImage Provides an R class for storing 2D and 3D images. It provides an interface to the ImageMagick image processing and analysis library. All R functions that work on numeric vectors and arrays can be applied to the image objects. Other methods focus on detecting objects (cells) in the images and extracting numeric descriptors for the objects, so they can be subjected to regression and machine learning methods.

The *BiocViews* system (see the **biocViews** package), of package categorization introduced last May has been well received and is now the primary web interface for browsing Bioconductor packages. The views for the 1.9 release are available at http://bioconductor.org/packages/1.9/BiocViews.html. We are exploring web application frameworks, in particular Django, for providing more flexible query-based package discovery. A link to our prototype package search application is available at http://bioconductor.org/apptest.

The 2.0 release of Bioconductor is scheduled for April 2007. Core members are focusing on the following milestones for the 2.0 release:

- A new generation of annotation data packages that use SQLite databases instead of R environment objects.

- Updates to the **annotate** package to provide more flexible interfaces to the SQLite-based annotation packages.

- The release of **oligo** which will replace the **affy** package and provide cross-manufacturer support (presently Affymetrix and Nimblegen) for new high-throughput technologies such as SNP, exon, and tiling arrays.

Seth Falcon
Program in Computational Biology
Fred Hutchinson Cancer Research Center
sfalcon@fhcrc.org

R News is a publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are copyrighted by the respective authors. Please send submissions to regular columns to the respective column editor and all other submissions to the editor-in-chief or another member of the editorial board. More detailed submission instructions can be found on the R homepage.

R Project Homepage:
http://www.R-project.org/

This newsletter is available online at
http://CRAN.R-project.org/doc/Rnews/